Bu kitaba sığmayan daha neler var!

Karekodu okut, bu kitapla ilgili EBA içeriklerine ulaş!

ISBN: 978-975-11-6250-2

BU DERS KİTABI MİLLÎ EĞİTİM BAKANLIĞINCA
ÜCRETSİZ OLARAK VERİLMİŞTİR.
PARA İLE SATILAMAZ.

Bandrol Uygulamasına İlişkin Usul ve Esaslar Hakkında Yönetmeliğin Beşinci Maddesinin İkinci Fıkrası Çerçevesinde Bandrol Taşıması Zorunlu Değildir.

BİLİŞİM TEKNOLOJİLERİ ALANI **NESNE TABANLI PROGRAMLAMA 10 DERS KİTABI**

MESLEKÎ VE TEKNİK ANADOLU LİSESİ

BİLİŞİM TEKNOLOJİLERİ 10
NESNE TABANLI PROGRAMLAMA
DERS KİTABI

T.C. MİLLÎ EĞİTİM BAKANLIĞI

MESLEKİ VE TEKNİK ANADOLU LİSESİ
BİLİŞİM TEKNOLOJİLERİ ALANI

NESNE TABANLI PROGRAMLAMA 10
DERS KİTABI

Yazarlar
Abdullah HOCAOĞLU
Devrim ALTINKURT
Hamza BALCI
Murat İMSİYATOĞLU
Mustafa NACAR
Yasemin AKPINAR

DEVLET KİTAPLARI

MİLLÎ EĞİTİM BAKANLIĞI YAYINLARI 8016
YARDIMCI VE KAYNAK KİTAPLAR DİZİSİ 1944

Her hakkı saklıdır ve Millî Eğitim Bakanlığına aittir. Kitabın metin, soru ve şekilleri kısmen de olsa hiçbir surette alınıp yayımlanamaz.

HAZIRLAYANLAR

Dil Uzmanı
Melek DEMİR

Program Geliştirme Uzmanı
Esra YAVUZ

Rehberlik Uzmanı
Gülşen YALIN

Ölçme ve Değerlendirme Uzmanı
Arzu DURSUN URGUN

Görsel Tasarım Uzmanı
Fırat DOĞAN
Özden ALTUN

ISBN: 978-975-11-6250-2

Millî Eğitim Bakanlığının 24.12.2020 gün ve 18433886 sayılı oluru ile Meslekî ve Teknik Eğitim Genel Müdürlüğünce ders materyali olarak hazırlanmıştır.

İSTİKLÂL MARŞI

Korkma, sönmez bu şafaklarda yüzen al sancak;
Sönmeden yurdumun üstünde tüten en son ocak.
O benim milletimin yıldızıdır, parlayacak;
O benimdir, o benim milletimindir ancak.

Çatma, kurban olayım, çehreni ey nazlı hilâl!
Kahraman ırkıma bir gül! Ne bu şiddet, bu celâl?
Sana olmaz dökülen kanlarımız sonra helâl.
Hakkıdır Hakk'a tapan milletimin istiklâl.

Ben ezelden beridir hür yaşadım, hür yaşarım.
Hangi çılgın bana zincir vuracakmış? Şaşarım!
Kükremiş sel gibiyim, bendimi çiğner, aşarım.
Yırtarım dağları, enginlere sığmam, taşarım.

Garbın âfâkını sarmışsa çelik zırhlı duvar,
Benim iman dolu göğsüm gibi serhaddim var.
Ulusun, korkma! Nasıl böyle bir imanı boğar,
Medeniyyet dediğin tek dişi kalmış canavar?

Arkadaş, yurduma alçakları uğratma sakın;
Siper et gövdeni, dursun bu hayâsızca akın.
Doğacaktır sana va'dettiği günler Hakk'ın;
Kim bilir, belki yarın, belki yarından da yakın.

Bastığın yerleri toprak diyerek geçme, tanı:
Düşün altındaki binlerce kefensiz yatanı.
Sen şehit oğlusun, incitme, yazıktır, atanı:
Verme, dünyaları alsan da bu cennet vatanı.

Kim bu cennet vatanın uğruna olmaz ki feda?
Şüheda fışkıracak toprağı sıksan, şüheda!
Cânı, cânânı, bütün varımı alsın da Huda,
Etmesin tek vatanımdan beni dünyada cüda.

Ruhumun senden İlâhî, şudur ancak emeli:
Değmesin mabedimin göğsüne nâmahrem eli.
Bu ezanlar -ki şehadetleri dinin temeli-
Ebedî yurdumun üstünde benim inlemeli.

O zaman vecd ile bin secde eder -varsa- taşım,
Her cerîhamdan İlâhî, boşanıp kanlı yaşım,
Fışkırır ruh-ı mücerret gibi yerden na'şım;
O zaman yükselerek arşa değer belki başım.

Dalgalan sen de şafaklar gibi ey şanlı hilâl!
Olsun artık dökülen kanlarımın hepsi helâl.
Ebediyyen sana yok, ırkıma yok izmihlâl;
Hakkıdır hür yaşamış bayrağımın hürriyyet;
Hakkıdır Hakk'a tapan milletimin istiklâl!

Mehmet Âkif Ersoy

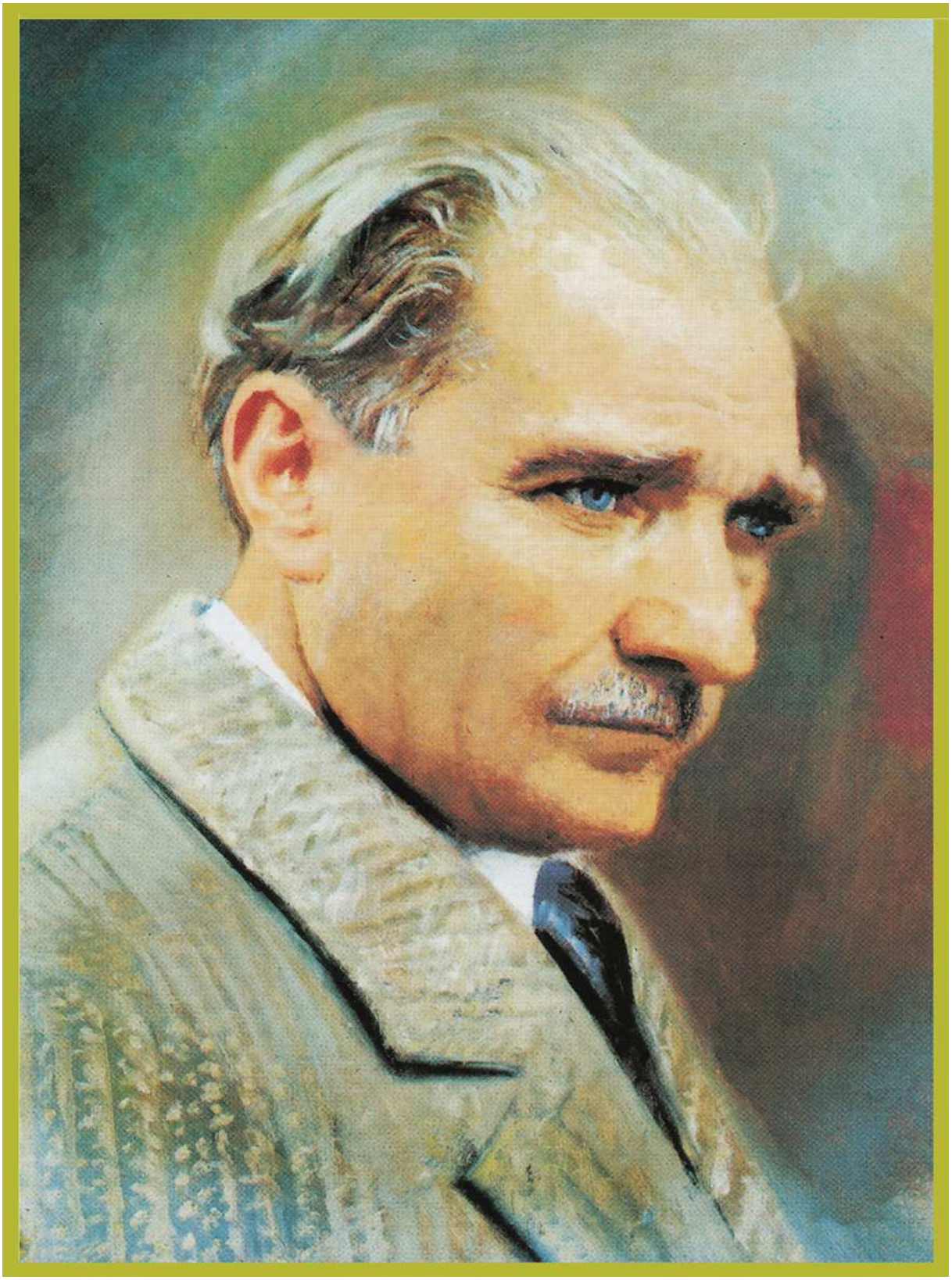
GENÇLİĞE HİTABE

Ey Türk gençliği! Birinci vazifen, Türk istiklâlini, Türk Cumhuriyetini, ilelebet muhafaza ve müdafaa etmektir.

Mevcudiyetinin ve istikbalinin yegâne temeli budur. Bu temel, senin en kıymetli hazinendir. İstikbalde dahi, seni bu hazineden mahrum etmek isteyecek dâhilî ve hâricî bedhahların olacaktır. Bir gün, istiklâl ve cumhuriyeti müdafaa mecburiyetine düşersen, vazifeye atılmak için, içinde bulunacağın vaziyetin imkân ve şeraitini düşünmeyeceksin! Bu imkân ve şerait, çok namüsait bir mahiyette tezahür edebilir. İstiklâl ve cumhuriyetine kastedecek düşmanlar, bütün dünyada emsali görülmemiş bir galibiyetin mümessili olabilirler. Cebren ve hile ile aziz vatanın bütün kaleleri zapt edilmiş, bütün tersanelerine girilmiş, bütün orduları dağıtılmış ve memleketin her köşesi bilfiil işgal edilmiş olabilir. Bütün bu şeraitten daha elîm ve daha vahim olmak üzere, memleketin dâhilinde iktidara sahip olanlar gaflet ve dalâlet ve hattâ hıyanet içinde bulunabilirler. Hattâ bu iktidar sahipleri şahsî menfaatlerini, müstevlîlerin siyasî emelleriyle tevhit edebilirler. Millet, fakr u zaruret içinde harap ve bîtap düşmüş olabilir.

Ey Türk istikbalinin evlâdı! İşte, bu ahval ve şerait içinde dahi vazifen, Türk istiklâl ve cumhuriyetini kurtarmaktır. Muhtaç olduğun kudret, damarlarındaki asil kanda mevcuttur.

Mustafa Kemal Atatürk

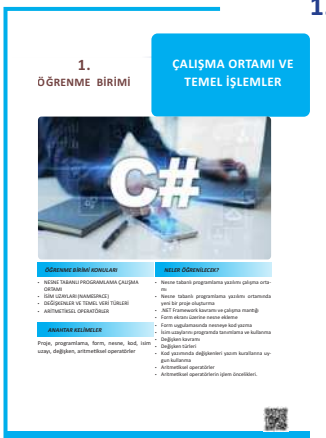


MUSTAFA KEMAL ATATÜRK

İÇİNDEKİLER

KİTABIN TANITIMI 13

1. ÖĞRENME BİRİMİ: ÇALIŞMA ORTAMI VE TEMEL İŞLEMLER 16



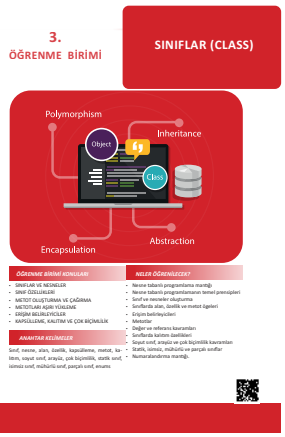
1.1. NESNE TABANLI PROGRAMLAMA ÇALIŞMA ORTAMI 16
1.2. C# PROGRAMLAMA DİLİ 16
1.3. .NET FRAMEWORK 17
1.3.1. C# ve .NET Framework İlişkisi 17
1.3.2. .NET Framework Çalışma Mantığı 18
1.4. KOD EDİTÖRÜ ARAYÜZ EKRANI 19
1.4.1. Form Ekranı 19
1.4.2. Araç Kutusu (Toolbox) 20
1.4.3. Özellikler (Properties) 21
1.4.4. Olaylar (Events) 22
1.4.5. Çözüm Penceresi (Solution Explorer) 23
1.4.6. Hata Listesi (Error List) 24
1.5. İSİM UZAYLARI (Namespace) 30
1.6. DEĞİŞKENLER VE TEMEL VERİ TÜRLERİ 31
1.6.1. Temel Veri Türleri 31
1.6.2. Değişken Tanımlama 32
1.6.3. Değişkene Değer Atama 33
1.6.4. Değişken İsimlendirme Kuralları 33
1.6.5. Değişken Veri Türü Dönüştürme (Convert) İşlemleri 34
1.7. ARİTMETİKSEL OPERATÖRLER 36
1.8. İŞLEM ÖNCELİĞİ 37

ÖLÇME VE DEĞERLENDİRME 41

2. ÖĞRENME BİRİMİ: KARAR VE DÖNGÜ YAPILARI 44

2.1. KARAR İFADELERİ 44
2.1.1. Karşılaştırma Operatörleri 44
2.1.2. if Yapısı 44
2.1.3. if-else Yapısı 47
2.1.4. else if Yapısı 49
2.1.5. İç İçe Şart İfadeleri 50
2.1.6. Switch-Case 52
2.2. MANTIKSAL OPERATÖRLER 53
2.2.1. AND (&&) Operatörü 54
2.2.2. OR(||) Operatörü 56
2.2.3. Mantıksal Operatör Önceliği 57
2.2.4. NOT(!) Operatörü 58
2.3. DÖNGÜLER 60
2.3.1. Sayaçlar 60
2.3.2. Artırma ve Azaltma Operatörleri 61
2.3.3. For Döngüsü 61
2.3.4. While Döngüsü 66
2.3.5. Do-while Döngüsü 67
2.3.6. Döngüyü Kesme (Durdurma) 68
2.3.7. Döngüyü Devam Ettirme 69
2.4. HATA AYIKLAMA 70
2.4.1. Try-Catch-Finally Bloku 70

ÖLÇME VE DEĞERLENDİRME 73



3. ÖĞRENME BİRİMİ: SINIFLAR (CLASS) 76

3.1. NESNE TABANLI PROGRAMLAMAYA GİRİŞ 76
3.1.1. NTP Öncesi 76
3.1.2. NTP Temel Prensipleri 76
3.2. SINIFLAR VE NESNELER 77
3.2.1. Sınıf Tanımlama 77
3.2.2. Nesne Oluşturma 78
3.3. KAPSÜLLEME, ALANLAR VE ÖZELLİKLER (ENCAPSULATION, FIELDS, PROPERTIES) 79
3.4. ERİŞİM BELİRLEYİCİLER (ACCESS MODIFIERS) 79
3.5. ALANLAR (FIELDS) 80
3.6. ÖZELLİKLER (PROPERTIES) 81
3.6.1. Sadece Okunabilir Özellikler 83
3.6.2. Sadece Yazılabilir Özellikler 84
3.7. METOTLAR (METHODS) 85
3.7.1. Varsayılan Değerli Parametreler (Optional Parameters) 87
3.7.2. İsimlendirilmiş Parametreler (Named Parameters) 87
3.7.3. Parametre Dizileri 88
3.7.4. Metodu Sonlandırma 88
3.7.5. Metot Aşırı Yüklemeleri (Method Overloads) 89
3.8. YAPICI VE YIKICI METOTLAR 91
3.8.1. Yapıcı Metotlar (Constructors) 91
3.8.2. Yıkıcı Metotlar (Destructors) 94
3.9. DEĞER VE REFERANS TİPLER 95
3.9.1. Metotlarda ref ve out Kullanımı 97
3.10. KALITIM (INHERITANCE) 99
3.10.1. Hiyerarşik Kalıtım 101
3.10.2. new Operatörüyle Metot Gölgeleme (Shadowing) 101
3.10.3. Sanal Metotlar (Virtual Methods) 102
3.11. SOYUT SINIFLAR (ABSTRACT CLASSES) 104
3.12. ARAYÜZLER (INTERFACES) 106
3.13. ÇOK BİÇİMLİLİK (POLYMORPHISM) 108
3.14. STATİK SINIFLAR (STATIC CLASSES) 110
3.15. İSİMSİZ SINIFLAR (ANONYMOUS CLASSES) 113
3.16. MÜHÜRLÜ SINIFLAR (SEALED CLASSES) 114
3.17. PARÇALI SINIFLAR (PARTIAL CLASSES) 114
3.18. ENUMS (NUMARALANDIRMALAR) 115

ÖLÇME VE DEĞERLENDİRME 117

4. ÖĞRENME BİRİMİ:DİZİLER (ARRAYS) VE KOLEKSİYONLAR (COLLECTIONS) 120

4.1. DİZİLER 120
4.1.1. Tek Boyutlu Diziler 120
4.1.2. Bir Boyutlu Dizilerin Oluşturulması 120
4.1.3. Bir Boyutlu Dizilere Değer Aktarma 121
4.1.4. Bir Boyutlu Dizi Elemanlarına Erişim 124
4.1.5. Dizilerde Foreach Döngüsü Kullanımı 125
4.1.6. Bir Boyutlu Dizilerde Kullanılan Özellikler ve Metotlar 130
4.1.7. Çok Boyutlu Diziler 131
4.1.8. İki Boyutlu Dizi Tanımlama 131
4.1.9. İki Boyutlu Diziye Değer Aktarma 132
4.1.10. İki Boyutlu Dizi Elemanlarına Erişim 134
4.2. KOLEKSİYONLAR 138
4.2.1. Boxing (Kutulama)-Unboxing (Kutu Açma) 139
4.2.2. ArrayList Koleksiyonu 139

4.2.3. List Koleksiyonu 145
4.2.4. Queue-Stack Koleksiyonları 147
4.2.5. Dictionary Koleksiyonu 150
4.2.6. Hashtable Koleksiyonu 152
4.2.7. SortedList Koleksiyonu 154
ÖLÇME VE DEĞERLENDİRME 155

5. ÖĞRENME BİRİMİ: FORM UYGULAMALARI 160
5.1. FORMLAR 160
5.1.1. Form Sınıfı 160
5.1.2. Kontrol Sınıfı 164
5.1.3. Konteyner Kontrolleri 166
5.2. MENÜLER 170
5.2.1. MenuStrip Kontrolü 170
5.2.2. ContextMenuStrip Kontrolü 173
5.3. İLETİŞİM KUTULARI (DIALOG BOXES) 174
5.3.1. Mesaj İletişim Kutusu (MessageBox) 175
5.3.2. Dosya Kaydet İletişim Kutusu (SaveFileDialog) 176
5.3.3. Dosya Aç İletişim Kutusu (OpenFileDialog) 177
5.3.4. Yazdırma İletişim Kutusu (PrintDialog) 178
5.3.5. Yazı Tipi İletişim Kutusu (FontDialog) 179
5.3.6. Renk İletişim Kutusu (ColorDialog) 179
5.4. VERİ DOĞRULAMA (VALIDATION) 180
5.4.1. İpucu (ToolTip) 180
5.4.2. Veri Girişi Doğrulama (Input Validation) 181
5.4.3. Veri Girişi Maskeleme (MaskedTextBox) 184
5.5. VERİ BAĞLAMA (DATA BINDING) 186
5.5.1. Basit Veri Bağlama (Simple Data Binding) 186
5.5.2. Kompleks Veri Bağlama (Complex Data Binding) 187
ÖLÇME VE DEĞERLENDİRME 193

6. ÖĞRENME BİRİMİ: VERİ TABANI İŞLEMLERİ 196
6.1. VERİ TABANI YAZILIMININ KURULUMU 196
6.1.1. Veri Tabanı (Database) 196
6.1.2. Veri Tabanı Yönetim Sistemi (Database Managament System) 196
6.1.3. MySQL Veri Tabanı Yazılımının Kurulumu 197
6.1.4. Veri Tabanı Arayüz Ekranı 207
6.1.5. SQL (Structured Query Language) 208
6.2. VERİ TABANI TASARIMI 209
6.2.1. Normalizasyon 210
6.2.2. Veri Türleri 213
6.2.3. Veri Tabanı Oluşturma 214
6.2.4. Veri Tabanında Anahtarlar (Keys) ve İndeksler 215
6.3. TABLO İŞLEMLERİ 217
6.3.1. Tablo Oluşturma 218
6.3.2. Tablolara Veri Girişi 221
6.4. SQL KOMUTLARI 222
6.4.1. INSERT INTO Komutu (Kayıt Ekleme) 223
6.4.2. SELECT Komutu (Verileri Listeleme) 224
6.4.3. Karşılaştırma Operatörleri 226

6.4.4. WHERE Şart İfadesi 226
6.4.5. Mantıksal Operatörler 227
6.4.6. Hesaplama Fonksiyonları 228
6.4.7. LIKE Komutu (Arama Operatörü) 230
6.4.8. Order By Komutu (Sıralama) 231
6.4.9. UPDATE Komutu (Veri Güncelleme) 231
6.4.10. DELETE Komutu (Veri Silme) 233
6.5. İLİŞKİSEL VERİ TABANI (RELATIONAL DATABASE) 233
6.5.1. İlişkisel Veri Tabanı Tasarımı 233
6.5.2. Tablolar Arası İlişkiler 235
6.5.3. İlişkisel Veri Tabanı Tablolarına Veri Girişi Yapılması 242
6.5.4. İlişkisel Veri Tabanında Sorgular 244
6.6. MySQL VERİ TABANININ YEDEĞİNİ ALMA VE GERİ YÜKLEME 246
6.7. NESNE TABANLI PROGRAMLAMADA VERİ TABANI KULLANIMI 247
6.7.1. ADO.NET (ActiveX Data Objects.NET) 247
6.7.2. DataGridView Bileşeni 249
6.8. KÜTÜPHANE OTOMASYONU PROJESİNİN GELİŞTİRİLMESİ 250
6.8.1. Windows Form Projesinin Oluşturulması 250
6.8.2. Veri Tabanı Bağlantı Sınıfının Oluşturulması 251
6.8.3. Proje Ana Sayfasının Hazırlanması 252
6.8.4. Öğrenci İşlemleri Sayfasının Hazırlanması 253
6.8.5. Kitap Tür İşlemleri Sayfasının Hazırlanması 258
6.8.6. Kitap İşlemleri Sayfasının Hazırlanması 262
6.8.7. Ödünç Kitap İşlemleri Sayfasının Hazırlanması 268
6.8.8. Kurulum (Setup) Hazırlanması 274
6.9. ORM YAPISI VE ENTITY FRAMEWORK 281
ÖLÇME VE DEĞERLENDİRME 293

KAYNAKÇA 295
GENEL AĞ KAYNAKÇASI 295
GÖRSEL KAYNAKÇA 295
CEVAP ANAHTARLARI 296

KİTABIN TANITIMI

HAZIRLIK ÇALIŞMALARI

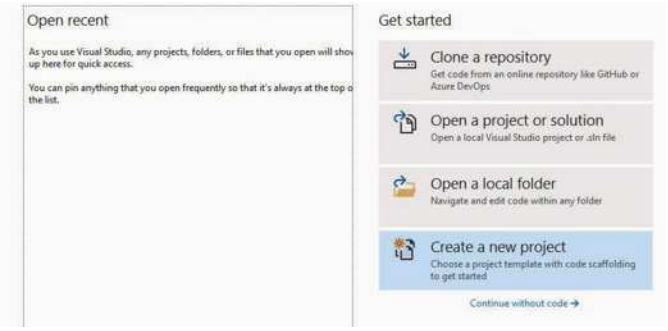
1. Programlama dili size ne ifade ediyor?
2. Bilgisayarda form yapısı ve pencere şeklinde kullanılan programlara örnekler veriniz.

Öğrenme biriminin başında yapılacak ön çalışmaları gösterir.

1. Uygulama

Konuları pekiştirmek için yapılan uygulamaları gösterir.

1. Adım: Kod editörü arayüzünü açarak Görsel 1.5'te görülen başlangıç ekranından Create a new project seçiniz ve yeni bir proje oluşturunuz.



Yapılan uygulamaları pekiştirme amaçlı kullanılan görselleri gösterir.

Görsel 1.1: Yeni proje oluşturma

Kullanılan görsel isimlerini ve görsel numaralarını gösterir.

Sıra Sizde

1. Toolbox panelinde **Label** ve **TextBox** nesnelerini çift tıklayarak bu nesneleri formun üzerine yerleştiriniz.
2. **Label** ve **TextBox** nesnelerini formun ortasına yerleştiriniz.

Konuları pekiştirmek için öğrencinin yapması gereken çalışmaları gösterir.

```
private void radioButton1_CheckedChanged(object sender, EventArgs e)
        {  pictureBox1.Visible = radioButton1.Checked; }
private void radioButton2_CheckedChanged(object sender, EventArgs e)
        {  pictureBox2.Visible = radioButton2.Checked; }
private void radioButton3_CheckedChanged(object sender, EventArgs e)
        {  pictureBox3.Visible = radioButton3.Checked; }
private void radioButton4_CheckedChanged(object sender, EventArgs e)
        {  pictureBox4.Visible = radioButton4.Checked; }
```

Kod panelini gösterir.

Not Bir sayı başka bir sayıya bölündüğünde kalan sayıya **mod** denir.

Konu hakkında dikkat çekici bilgileri gösterir.

Örnek

```
int sayi,deger,sonuç;
string isim1,isim2,soyad;
```

Konuları pekiştirmek için verilen örnekleri gösterir.

ÖLÇME VE DEĞERLENDİRME

Ölçme ve değerlendirme sorularını gösterir.

KİTABIN TANITIMI

Öğrenme biriminin sırasını gösterir.

Öğrenme biriminin adını gösterir.

Öğrenme biriminin kapak resmini gösterir.

Öğrenme birimindeki ana kazanımları gösterir.

Öğrenme biriminin karekodunu gösterir.

Tablo 3.1: Aynı Referansa Sahip Nesneler

Konuları pekiştirmek için kullanılan tablo isimlerini gösterir.

ref	out
Metodu tanımlarken parametrenin önüne “ref” yazılmalıdır.	Metodu tanımlarken parametrenin önüne “out” yazılmalıdır.
Metodu çağırırken değişkenin önüne “ref” yazılmalıdır.	Metodu çağırırken değişkenin önüne “out” yazılmalıdır.
Metoda göndermeden önce değişken başlangıç değeri almak zorundadır.	Metoda göndermeden önce değişken başlangıç değeri almak zorunda değildir.
Metot içinde istenildiği gibi kullanılabilir.	Metot içinde mutlaka bir değer ataması gerçekleştirilmelidir.

VERİ TABANI İŞLEMLERİ

Öğrenme birimi üst resmini gösterir.

Öğrenme birimi adını gösterir.

Ders adını gösterir.

Sayfa numarasını gösterir.

1.
ÖĞRENME BİRİMİ

ÇALIŞMA ORTAMI VE TEMEL İŞLEMLER

KONULAR

1.1. NESNE TABANLI PROGRAMLAMA ÇALIŞMA ORTAMI

1.2. İSİM UZAYLARI (NAMESPACE)

1.3. DEĞİŞKENLER VE TEMEL VERİ TÜRLERİ

1.4. ARİTMETİKSEL OPERATÖRLER

ANAHTAR KELİMELER

Proje, programlama, form, nesne, kod, isim uzayı, değişken, aritmetiksel operatörler

NELER ÖĞRENECEKSİNİZ?

- Nesne tabanlı programlama yazılımı çalışma ortamı
- Nesne tabanlı programlama yazılımı ortamında yeni bir proje oluşturma
- .NET Framework kavramı ve çalışma mantığı
- Form ekranı üzerine nesne ekleme
- Form uygulamasında nesneye kod yazma
- İsim uzaylarını programda tanımlama ve kullanma
- Değişken kavramı
- Değişken türleri
- Kod yazımında değişkenleri yazım kurallarına uygun kullanma
- Aritmetiksel operatörler
- Aritmetiksel operatörlerin işlem öncelikleri

HAZIRLIK ÇALIŞMALARI

1. Programlama dili size ne ifade ediyor?
2. Bilgisayarda form yapısı ve pencere şeklinde kullanılan programlara örnekler veriniz.

1.1. NESNE TABANLI PROGRAMLAMA ÇALIŞMA ORTAMI

Program, herhangi bir elektronik cihaza bir işlem yaptırmak için yazılan komutlar dizisidir. Programlar; bilgisayar, cep telefonu, tablet, elektronik ev eşyaları, araba ve daha birçok yerde kullanılır.

Elektronik cihazlara bilgisayar, cep telefonu, akıllı saat, akıllı televizyon vb. örnek olarak verilebilir (Görsel 1.1). Program yazmak denilince akla ilk gelen bilgisayarlar olsa da günümüzde pek çok elektronik cihaza kod yazılabilir. Bilgisayar ve diğer elektronik cihazlar çalışma prensibi olarak programları kullanır. Bu cihazlar önce girilen bilgiyi alır, ardından bu bilgiyi işler ve en sonunda ortaya bir sonuç çıkarır. Bu sonuç bazen bir mesaj bazen de bir işlemi gerçekleştirme şeklindedir. Bu durum, insanların birbiriyle iletişimi gibi düşünülebilir. İnsanlar da konuşmalarında önce karşı tarafın ifadelerini algılar, algıladıklarını işler ve son olarak karşı tarafa bir cevap verir.

Görsel 1.1: Nesne tabanlı programlama çalışma ortamı

Programlar önceden "1" ve "0"lardan oluşan kod blokları ile makine dilinde yazılırdı. Makine dilinde kod yazılması ve yazılan kodun anlaşılması oldukça zordu. Bu nedenle daha okunaklı ve kolay kod yazılabilen Assembly dilleri geliştirildi. İlerleyen zamanlarda daha çok anlaşılır, konuşma diline daha yakın ifadelerle kod yazma imkânı sağlayan C, C++, Turbo Pascal, Visual Basic, C#, Python gibi programlama dilleri kullanıldı.

Programlama; bilgisayar programlarının yazılması, test edilmesi ve bakımının yapılması sürecine verilen isimdir. Bu sürecin daha verimli geçmesi için amaca yönelik bir programlama dili seçilmelidir. Bu kitabın içeriğinde C# programlama dilinin kullanımı anlatılacaktır.

1.2. C# PROGRAMLAMA DİLİ

C# programlama dili, nesne tabanlı olarak geliştirilmiş bir dildir. Günlük hayatın birçok alanında büyük küçük pek çok şirket C# ile geliştirilmiş programları kullanılır (Görsel 1.2).

C# programlama dili ile;

- Mobil uygulamalar,
- Konsol uygulamaları,
- Web servisleri,
- Dinamik kütüphaneler (DLL),
- Oyun tasarımı,
- Form uygulamaları yapılabilir.

Görsel 1.2: Program çalışma süreci

C# programlama dilinin çok tercih edilmesinin sebeplerinden bazıları şunlardır:

- Yazılması ve anlaşılması kolay kod yapısına sahiptir.
- Yeni teknolojileri destekler.
- Kullanışlıdır.
- Ekip çalışmasına elverişlidir.
- Kullanıcıyla etkileşimlidir.
- Grafik arayüzlü tasarımlar yapılabilir.
- Ağ üzerinden birbiriyle uyumlu çalışabilir.
- Çevrimiçi veya çevrimdışı kullanılabilir.
- QR kod okuyucu, kamera, yazıcı vb. cihazlarla etkileşimlidir.
- Verileri depolayıp işleyerek analiz yapabilir.
- Sosyal medya platformları ile etkileşimlidir.
- Cep telefonu uygulamaları ile etkileşimlidir.
- Yapay zekâ teknolojisi kullanılarak yüz tanıma, nesne tanıma, ses tanıma işlemleri yapılabilir.
- Birçok özelliğe sahip farklı programlar geliştirebilme imkânı sağlar.

1.3. .NET FRAMEWORK

Framework kelimesi **geliştirme çatısı** anlamına gelir. .NET Framework, çoğu kişi tarafından bir programlama dili sanılır fakat programlama dillerinden bağımsız ve farklı programlama dilleri ile ortak çalışma imkânı sağlayan bir uygulama geliştirme platformudur. .NET logosu Görsel 1.3'te görülür. Farklı dilleri bilen programcılar ortak bir projeyi yürütebilir. .NET Framework çalışma mantığı bunun için uygundur. Yaygın kullanılan çoğu dil (C#, Visual Basic, Visual C++, Visual F#, Python) .NET Framework desteklidir. Ayrıca .NET Framework altyapısında kullanıma sunulmuş hazır kod kütüphaneleri ile kod yazma çok daha hızlı ve verimlidir. .NET Framework kütüphaneleri tüm .NET dillerinde ortak kullanılır. Bu yüzden .NET çok güçlü bir Framework'tür.

Görsel 1.3: .NET Framework

1.3.1. C# ve .NET Framework İlişkisi

C# bir programlama dilidir, .NET Framework ise C# dili ve birçok dilin kütüphanelerinin yüklü olduğu bir uygulama geliştirme platformudur. C# dilinde kullanılan kütüphanelerin tümü .NET Framework kütüphaneleridir.

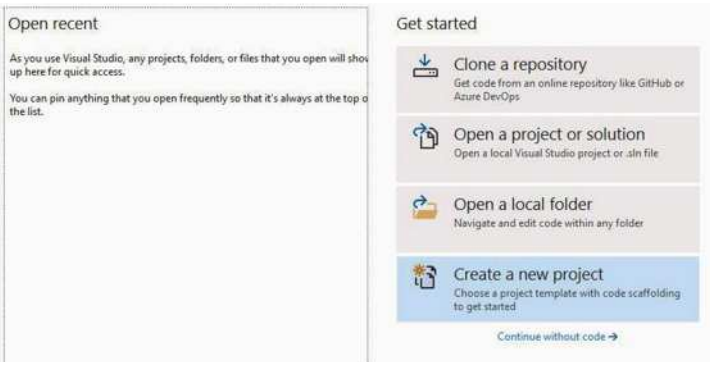
1.3.2. .NET Framework Çalışma Mantığı

Programlama dilleri ile yazılan kodlar makine için anlamlı değildir. Kodların makine dilinde yazılması veya makine diline çevrilmesi gerekir. Program yazılırken kullanılan kodlar derlendiğinde doğrudan makine diline çevrilmez. Görsel 1.4'te görüldüğü gibi kodlar önce **ara dil** (Intermediate Language) koduna, ardından da **çalışma zamanı derleyicisi** (Just-In-Time Compiler) tarafından makine diline çevrilir. Böylelikle kodlar sorunsuz bir şekilde çalışır. Bu işlemlerin yapılmasını .NET Framework altyapısı sağlar.

Görsel 1.4: .NET Framework çalışma mantığı

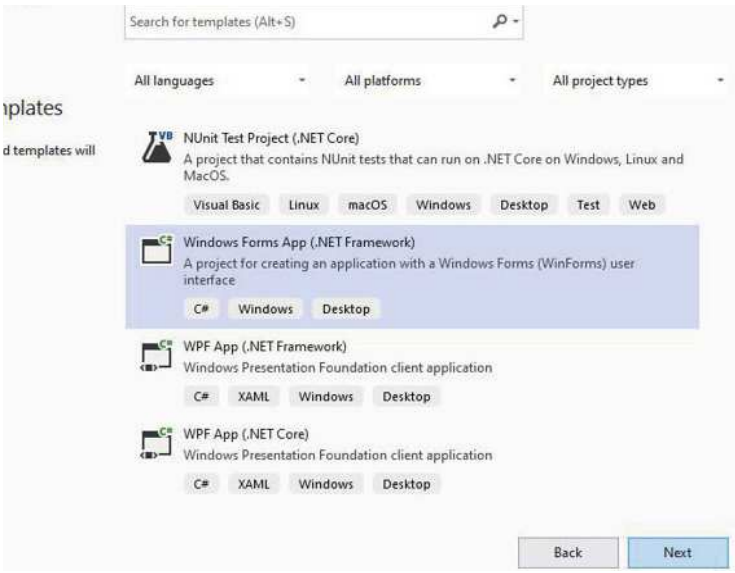
1. Uygulama

1. Adım: Kod editörü arayüzünü açarak Görsel 1.5'te görülen başlangıç ekranından Create a new project seçeneğini seçiniz ve yeni bir proje oluşturunuz.



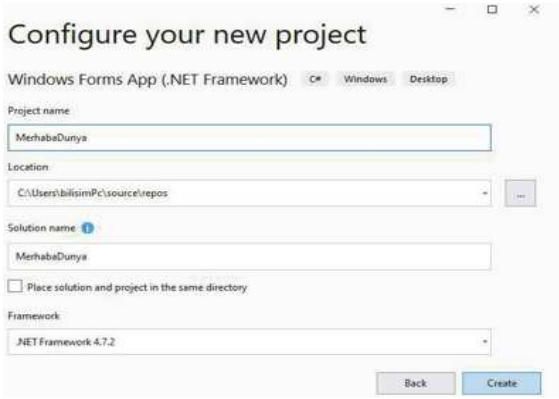
Görsel 1.5: Yeni proje oluşturma

2. Adım: Görsel 1.6'da görülen ekrandan Form Uygulamalarını (.NET Framework) seçiniz.



Görsel 1.6: Programlama dili seçimi

3. Adım: Projeye isim veriniz ve projenin kayıt yerini belirleyiniz (Görsel 1.7).



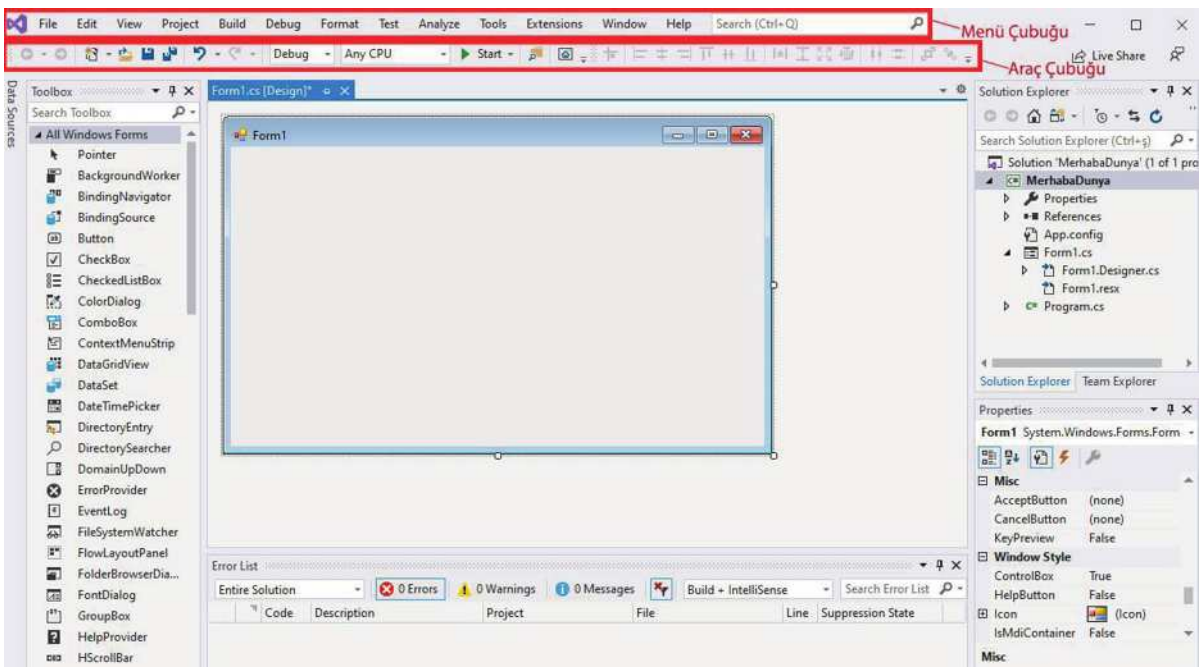
Görsel 1.7: Proje adı belirleme

Sıra Sizde

1. "MerhabaDunya" isimli bir Form uygulaması oluşturunuz.
2. Oluşturduğunuz "MerhabaDunya" isimli projeyi kaydedip kapattıktan sonra var olan projeyi açma seçeneği ile yeniden açınız.

1.4. KOD EDİTÖRÜ ARAYÜZ EKRANI

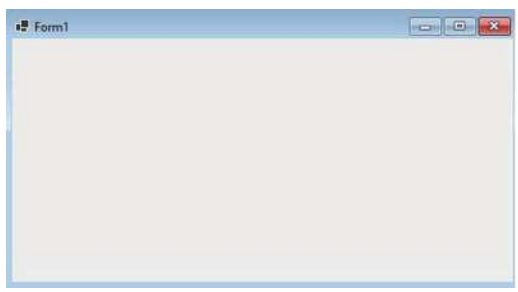
Görsel 1.8'de görüldüğü gibi kod editörü arayüz ekranı açıldığında ekranın üst tarafında Menü Çubuğu ve Araç Çubuğu, ekranın orta kısmında Form Ekranı ve form ekranının etrafında dört adet panel bulunur. Bu panellerin yerleri isteğe bağlı olarak sabitlenebilir, sürükle bırak yöntemi ile yerleri değiştirilebilir veya paneller tamamen kaldırılabilir.



Görsel 1.8: Kod editörü arayüz ekranı

1.4.1. Form Ekranı

Form ekranı, programın görsel tasarımının yapıldığı yerdir (Görsel 1.9). **Araç Kutusu (Toolbox)** panelinde bulunan nesneler form üzerinde istenilen pozisyona yerleştirilerek programın tasarımı yapılır.



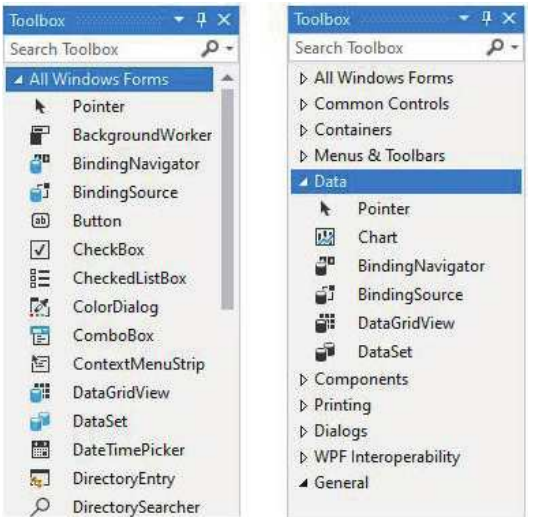
Görsel 1.9: Form ekranı

1.4.2. Araç Kutusu (Toolbox)

Form üzerinde tasarım için kullanılabilecek nesneler bu panelde bulunur. Görsel 1.10'da görüldüğü gibi tüm nesneler aynı anda veya işlevlerine göre çeşitli kategorilerde listelenebilir. Örneğin Data bölümüne tıklandığında **Data (Veri)** ile ilgili nesneler listelenir. Ayrıca **Search Toolbox (Arama Çubuğu)** ile istenilen nesnenin ismi yazılarak da nesne listelenebilir.

Toolbox paneli kullanılarak form üzerine nesne ekleme işlemi iki farklı şekilde yapılabilir:

1. Eklenecek nesne çift tıklanır.
2. Eklenecek nesne sürüklenip formun üzerinde herhangi bir pozisyona bırakılır.



Görsel 1.10: Araç kutusu

Sıra Sizde

1. Toolbox panelinde **Label** ve **TextBox** nesnelerini çift tıklayarak bu nesneleri formun üzerine yerleştiriniz.
2. **Label** ve **TextBox** nesnelerini formun ortasına yerleştiriniz.

En Çok Tercih Edilen Toolbox Nesneleri

Toolbox'ta 50'den fazla nesne bulunur. Bu nesnelerin büyük çoğunluğu Görsel 1.11'de ve Görsel 1.12'de görülür. Projelerde en çok kullanılan nesneler aşağıda listelenmiştir.

Button: Programlarda bazı kodları çalıştırmak için kullanılan komut düğmeleridir. Button nesnesine tıklandığında içeri doğru basma efekti gerçekleştiği için tıklama (Click) olayları için vazgeçilmez bir nesnedir.

CheckBox: Kullanıcıya bir veya aynı anda birden çok seçeneği işaretleme imkânı sağlayan nesnedir.

ComboBox: Açılır liste ile açılan seçenekler arasından seçim yapılmasına olanak sağlayan araçtır. Listeye yeni eleman ekleme ve çıkarma işlemleri, tasarım ekranından veya program çalışırken kod ile yapılabilir.

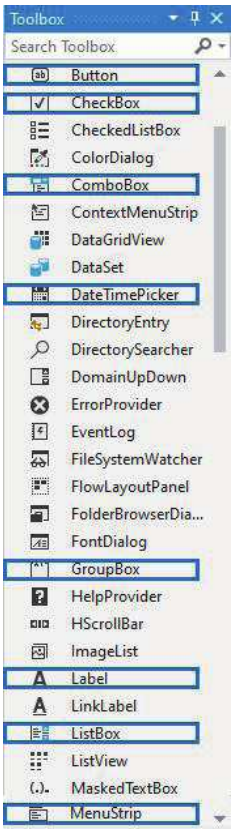
DateTimePicker: Tarih ve saat seçme işlemlerine olanak sağlayan nesnedir.

GroupBox: Form elemanlarını kendi aralarında gruplandırmak için kullanılan nesnedir. Nesneler gruplar hâlinde olduğu için daha anlaşılır tasarımlar yapılabilir.

Label: Form üzerinde bilgi vermek için kullanılan nesnedir.

ListBox: Sunulacak seçeneklerin açık bir liste hâlinde gösterildiği nesnedir.

MenuStrip: Programda menü başlıkları ve alt başlıkları oluşturmak için kullanılan nesnedir.



Görsel 1.11: En çok tercih edilen nesneler

PictureBox: Form üzerinde resim göstermek için kullanılan nesnedir.

ProgressBar: Yapılan bir işlemin ne kadarının tamamlandığını göstermek için kullanılan nesnedir.

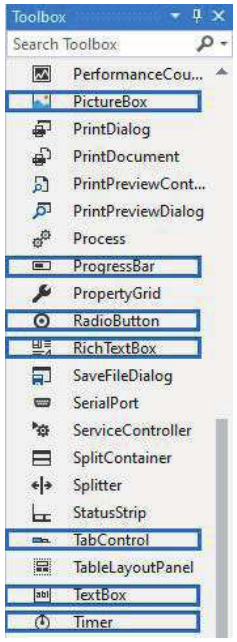
RadioButton: CheckBox nesnesinden farklı olarak birçok seçenek içinden sadece birinin seçilmesine imkân sağlayan nesnedir.

RichTextBox: Birden çok satır içine metin girişi yapılabilen nesnedir.

TabControl: Form elemanlarının gruplara ayrıldığı, grupların içindeki elemanları görmek için sekmelerin kullanıldığı nesnedir. Her sekme, bir grubu temsil eder.

TextBox: İçine tek satır metin girişi yapılabilen nesnedir. Bilgi girişi için en çok tercih edilen nesnedir.

Timer: Kodların zamanlanarak çalışmasını sağlayan nesnedir.



Görsel 1.12: En çok tercih edilen nesneler-2

Sıra Sizde

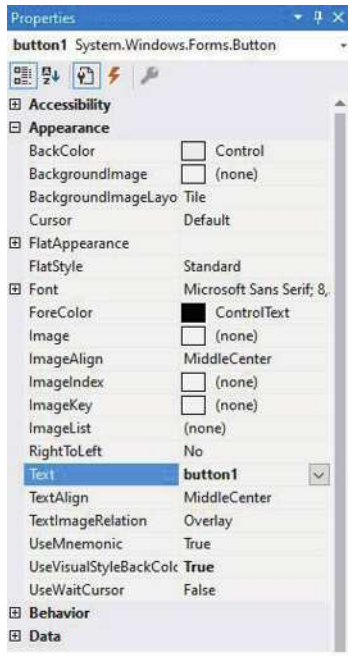
En çok tercih edilen nesnelerin özelliklerini araştırınız ve bulduğunuz özellikleri uygulama içinde kullanınız. Çalışmanızı yaparken öğretmeninizden destek alınız.

1.4.3. Özellikler (Properties)

Form nesnesinin ve diğer tüm nesnelerin özelliklerinin listelendiği, değiştirildiği ve ayrıca nesnelere ait **olayların (events)** listelendiği paneldir.

Her nesnenin kendine ait benzersiz özellikleri olduğu gibi diğer nesneler ile ortak özellikleri de bulunur. Tüm nesnelerin en temel ortak özelliği, isim (name) özelliğidir. Örneğin PictureBox nesnesinin genişlik ve yükseklik özelliği vardır fakat CheckBox nesnesinin yoktur. Button nesnesinin yazı rengi özelliği vardır fakat PictureBox nesnesinin yoktur. DateTimePicker nesnesinin tarih belirleme özelliği vardır fakat TextBox nesnesinin yoktur.

Görsel 1.13'te seçili button1 nesnesine ait bazı özellikler aşağıda verilmiştir.



Görsel 1.13: Özellikler sekmesi

BackColor: Nesnenin arka plan rengini değiştirir.

BackgroundImage: Nesnenin arka planına resim ekler.

Font: Nesnenin yazı tipini, boyutunu ve kalınlığını değiştirir.

ForeColor: Nesnenin yazı rengini değiştirir.

Text: Nesnenin yazı metnini değiştirir.

TextAlign: Nesnenin yazısını hizalar.

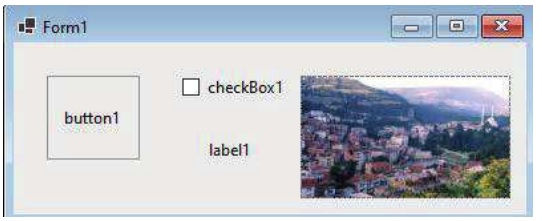
Sıra Sizde

Sizler de form üzerine iki farklı nesne ekleyip her nesnenin en az yedi özelliğini değiştiriniz.

2. Uygulama

1. Adım: İki kişilik gruplar hâlinde eşleşiniz. Yanınızdaki veya öğretmeninizin belirlediği bir arkadaşınız ile grup oluşturabilirsiniz.

2. Adım: Görsel 1.14'te görüldüğü gibi Form üzerine Button, CheckBox, Label ve PictureBox nesnelerini ekleyiniz. Toolbox panelinden sürükle bırak yöntemi ile nesneleri form üzerine yerleştiriniz.



Görsel 1.14: Özellikler paneli uygulama

3. Adım: Bu nesnelerin ortak ve farklı özelliklerini belirleyip defterinize not alınız. Form üzerindeki nesnelerin hepsi fare ile aynı anda seçildiğinde Properties panelinde nesnelerin sadece ortak özellikleri listelenir.

1.4.4. Olaylar (Events)

Her nesnenin form üzerinde bir görevi bulunur. Bazı nesneler sadece programın işlevi ile ilgili bilgiyi ve görseli yansıtmak için kullanılır. Bazı nesneler ise belli durumlarda (üzerine tıklandığında, bir tuşa basıldığında vs.) kod parçacıklarını çalıştırmak için kullanılır. Nesneler ile kullanıcı etkileşimi olaylar sayesinde sağlanır. Günlük hayatta sosyal medya uygulamalarında, web sitelerinde, oyunlarda ve daha birçok alanda nesnelere tanımlanmış olay metotları çalışarak kullanıcıyla etkileşim sağlanır. Olay metotları tanımlanarak nesnelerin hangi durumda, nasıl kodlar çalıştırabileceği belirlenir. Örneğin butona tıklandığında şifre kontrolünün yapılması, klavyeden sağ ok tuşuna basıldığında bir sonraki resmin gösterilmesi, PictureBox nesnesinin üzerine çift tıklama yapıldığında resme ait ilgili bilgilerin MessageBox ile göstermesi vb. Olaylar için metot tanımlanırken istenilen olayın adının hemen yanındaki boş beyaz kutucuğa çift tıklanır ve otomatik olarak aşağıdaki gibi bir metot oluşturulur.

```
private void button1_Click(object sender, EventArgs e)
                    { listBox1.Items.Add(textBox1.Text); }
```

Görsel 1.15: Olaylar sekmesi

Görsel 1.15'te seçili **button1** nesnesine ait bazı olaylar aşağıda verilmiştir.

Click: Nesneye fare ile tıklanması veya fare nesne üzerindeyken enter tuşuna basılmasıyla devreye giren olaydır.

MouseClick: Click gibi nesnenin fare ile tıklanması olayıdır fakat MouseClick, enter tuşundan etkilen-

mez. Ayrıca MouseClick, fareye ait koordinatlar gibi özel bilgileri de verir.

KeyDown: Klavyeden bir tuşa basılması olayıdır.

KeyUp: Klavyede basılan tuşun bırakılması olayıdır.

MouseDown: Farenin tuşuna basılması olayıdır.

Sıra Sizde

Form nesnesinin Load, Enter, Click olaylarına ait metotları oluşturunuz.

3. Uygulama

1. Adım: Form üzerine TextBox ve Button nesnelerini ekleyiniz.

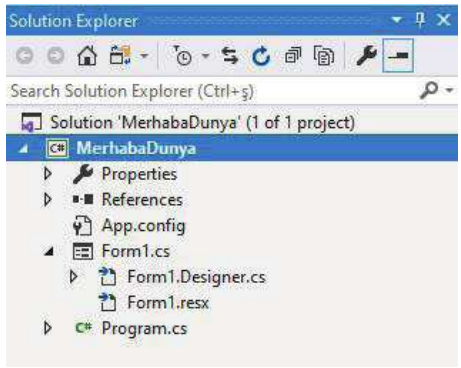
2. Adım: Eklediğiniz nesnelerin özeliklerini Görsel 1.16'daki gibi tasarıma uygun hâle getiriniz.

3. Adım: Form üzerindeki nesnelerin ortak olan olaylarını belirleyip sınıf arkadaşlarınızla paylaşınız. Form üzerindeki nesnelerin hepsi fare ile aynı anda seçildiğinde Properties panelinde nesnelerin sadece ortak olayları listelenir.

Görsel 1.16: Özellikler paneli uygulama-2

1.4.5. Çözüm Penceresi (Solution Explorer)

Proje ile ilgili tüm dosya ve klasörlerin listelenerek silme, kopyalama, taşıma, isim değiştirme işlemlerinin yapılabildiği paneldir (Görsel 1.17). Projenin detaylı bir haritası gibi düşünülebilir. Çözüm penceresi ile projeye yeni sınıf (class), form ve başka ögeler eklenebilir. Veri tabanı dosyası, resim, müzik, video dosyaları projeye dâhil edilebilir.



Görsel 1.17: Çözüm penceresi

Projeye eklenen her form, çözüm penceresinde ayrı ayrı listelenir. Listedeki Form1.cs ifadesine fare sağ tuşu ile tıklandığında açılan listeden View Code tıklanır ise Form1'e ait kod ekranı, View Designer tıklanır ise Form1'e ait tasarım ekranı açılır. Doğrudan Form1.cs ifadesine çift tıklanır ise tasarım ekranı açılır.

Kod veya tasarım ekranını göstermenin diğer yolları şunlardır:

- Form ekranı üzerinde herhangi bir noktada fare sağ tuşuyla açılan listede **View Code (Kodu Göster)** tıklanarak kod ekranı, **View Designer (Tasarımı Göster)** tıklanarak tasarım ekranı görüntülenir.
- Klavyeden **F7** tuşuna basıldığında kod ekranı açılır. Klavyeden **Shift + F7** tuşlarına basıldığında tasarım ekranı açılır.

1.4.6. Hata Listesi (Error List)

Kod yazarken, kod derlenirken veya kod çalışırken oluşan hataların ve uyarıların listelendiği Görsel 1.19'da görülen paneldir.

a) Errors (Hatalar) Bölümü: Çok kritik ve programın çalışmasını engelleyen hatalardır. Örneğin değişkeni tanımlamadan bir kod bloku içinde kullanmak, kod satırının sonunda noktalı virgül koymamak, kod yazarken açılan parantezin kapatılmaması vb.

b) Warnings (Uyarılar) Bölümü: Programın çalışmasını engellemeyecek düzeydeki iletilerdir. Örneğin değişkenin tanımlanıp hiçbir zaman kullanılmaması vb. Görsel 1.18'de hatalı bir kod bloku verilmiştir.

```
24
25      1 reference
        private void button1_Click(object sender, EventArgs e)
26      {
27          int a;
28          a="yazilim"
29      }
30  }
31  }
32
```

Görsel 1.18: Hatalı kod bloku

Birinci hata, 28. satırda bulunan kodun sonunda noktalı virgül olmamasıdır. İkinci hata ise 28. satırdaki kodda bulunan "a" isimli değişkenin tipi sayısal olmasına rağmen değişkene metinsel ifade atanmış olmasıdır. Hatalar kritik seviyede olduğu için program çalışmaz. Görsel 1.19'da görüldüğü gibi program derleyicisi bu iki hatayı yakalar ve Error List panelinde listeler.

Error List

Entire Solution | 2 Errors | 0 Warnings | 0 of 2 Messages | Build + IntelliSense | Search Error List

	Code	Description	Project	File	Line	Suppression State
⊗	CS1002	; expected	MerhabaDunya	Form1.cs	28	Active
⊗	CS0029	Cannot implicitly convert type 'string' to 'int'	MerhabaDunya	Form1.cs	28	Active

Görsel 1.19: Hata listesi

4. Uygulama

1. Adım: Form üzerine Görsel 1.20'de görüldüğü gibi bir tane Button nesnesi ekleyiniz.

2. Adım: Özellikler penceresinden formun Text özelliğini "Merhaba Dünya" yapınız.

Görsel 1.20: Merhaba dünya

3. Adım: Özellikler listesinden button1'in Text özelliğini mesaj göster yapınız.

4. Adım: Olaylar listesinden button1 için Click olayı metodunu button1'in üzerine çift tıklayarak oluşturunuz.

5. Adım: Oluşturduğunuz metodun içine MessageBox.Show ("Merhaba Dünya"); kodunu yazınız.

6. Adım: Araç Çubuğunda bulunan ▶ Başlat butonu ile programı çalıştırınız.

```
private void button1_Click(object sender, EventArgs e)
    {
        MessageBox.Show("Merhaba Dünya");
    }
```

Örnek uygulamada en önemli nokta, Click olayının metodunu oluşturmaktır. Click, programcıların en çok kullandığı olaydır ve bu olay için metot oluşturma işlemi en pratik şekilde seçili nesneye çift tıklanarak gerçekleştirilir. Uygulamada diğer olaylar için metot tanımlama işlemi istenilseydi Görsel 1.15'teki gibi button1'e ait olayların listesinden istenilen olayın adının hemen yanındaki boş kutucuğa çift tıklanarak metot oluşturulurdu.

Not

MessageBox, adından da anlaşılacağı gibi ekrana mesaj verdiren sınıfın adıdır. Programcılar tarafından çok fazla kullanılır. Farklı kullanım şekilleri vardır. En temel kullanımı, örnek kodda verildiği gibi sadece tek bir mesajı gösterme şeklindedir. MessageBox sınıfının bazı kullanım şekilleri aşağıda verilmiştir.

```
1. MessageBox.Show("mesaj metni","mesaj başlığı");
2. MessageBox.Show("mesaj metni","mesaj başlığı",MessageBoxButtons.YesNoCancel);
3. MessageBox.Show("mesaj metni","mesaj başlığı",MessageBoxButtons.OKCancel);
```

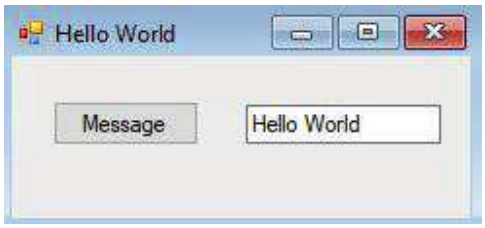
Sıra Sizde

MessageBox sınıfının farklı kullanım şekilleri neler olabilir? Araştırıp bulduklarınızı program içinde kullanınız ve arkadaşlarınızla paylaşınız.

5. Uygulama

1. Adım: Form üzerine Görsel 1.21'de görüldüğü gibi birer tane Button ve TextBox nesnesi ekleyiniz.

2. Adım: Button1'e tıklandığında textBox1 nesnesinin içine Hello World yazdırınız.



Görsel 1.21: Hello World mesaj uygulaması

```
private void button1_Click(object sender, EventArgs e)
{
textBox1.Text = "Hello World";
}
```

Yukarıdaki örnekte mesaj, bir önceki uygulamadan farklı olarak TextBox nesnesi üzerinden verilmiştir. **textBox1.Text="Hello World";** kod parçasına bakılırsa nesnenin bir özelliğine kod ile müdahale edildiği görülür. Nesnelerin özellikleri sadece Properties panelinden değiştirilmez. Programın çalışma zamanında kodla da değiştirilebilir.

Örnek kodlar aşağıda verilmiştir.

```
textBox1.ForeColor = Color.Red; //Yazı rengini kırmızı yapar.
textBox1.Enabled = false;  //Nesneyi pasifleştirir. Artık metin girişi yapılamaz.
textBox1.Visible = false;  //Nesneyi görünmez hâle getirir.
textBox1.Font    = new Font("Broadway", 16);  //Yazı tipi ve boyutu değişir.
```

6. Uygulama

http://kitap.eba.gov.tr/KodSor.php?KOD=21070

1. Adım: Form üzerine Görsel 1.22'de görüldüğü gibi iki tane GroupBox nesnesi ekleyiniz.

2. Adım: Üst tarafta bulunan GroupBox nesnesinin içine bir tane Label nesnesi ekleyiniz.

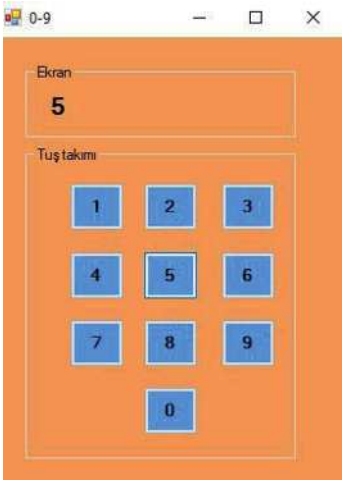
3. Adım: Alt tarafta bulunan GroupBox nesnesinin içine on tane Button nesnesi ekleyiniz ve bunları numaralandırınız.

4. Adım: Numaraların yazılı olduğu Button nesnelerinin arka plan rengini "MenuHighlight" yapınız.

5. Adım: Button nesnelerinin ve Label nesnesinin yazı tipi stilini "kalın", yazı boyutunu "10" yapınız.

6. Adım: Form nesnesinin başlığını "0-9" ve Form nesnesinin arka plan rengini "SandyBrown" yapınız.

7. Adım: Tıklanan Button nesnesine ait sayıyı Label nesnesinin Text özelliğine aktaran programı yazınız.



Görsel 1.22: Tuş takımı

```
private void button1_Click(object sender, EventArgs e)
            {  label1.Text = "1";  }
private void button2_Click(object sender, EventArgs e)
            {  label1.Text="2";  }
private void button3_Click(object sender, EventArgs e)
            {  label1.Text = "3";  }
private void button4_Click(object sender, EventArgs e)
            {  label1.Text = "4";  }
private void button5_Click(object sender, EventArgs e)
            {  label1.Text = "5";  }
private void button6_Click(object sender, EventArgs e)
            {  label1.Text = "6";  }
private void button7_Click(object sender, EventArgs e)
            {  label1.Text = "7";  }
private void button8_Click(object sender, EventArgs e)
            {  label1.Text = "8";  }
private void button9_Click(object sender, EventArgs e)
            {  label1.Text = "9";  }
private void button10_Click(object sender, EventArgs e)
            {   label1.Text = "0";
```

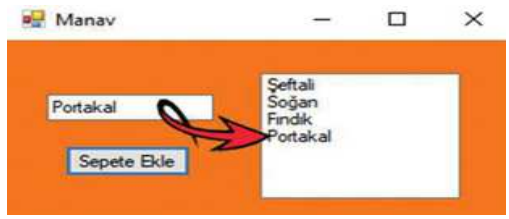
Sıra Sizde

1. Form üzerine 2 adet **Button** nesnesi ve 1 adet **PictureBox** nesnesi ekleyiniz.
2. Button1'e tıklandığında PictureBox nesnesini görünmez hâle getiren, button2'ye tıklandığında PictureBox nesnesini görünür hâle getiren programı yazınız.

7. Uygulama

Sepete Ekle butonuna tıklandığında TextBox nesnesindeki değeri ListBox nesnesine aktaran Görsel 1.23'teki gibi bir tasarıma sahip programı yazınız.

```
private void button1_Click(object sender, EventArgs e)
    {
        listBox1.Items.Add(textBox1.Text);
    }
```



Görsel 1.23: Manav sepeti uygulaması

Sıra Sizde

1. Görsel 1.23'teki program tasarımına **Sepeti Temizle** butonu ekleyiniz.
2. Yeni eklediğiniz butona tıklandığında ListBox nesnesinin içindekileri tamamen temizleyen programı yazınız.

Sıra Sizde

1. Form üzerine 2 adet **Button** nesnesi ekleyiniz.
2. Button1'e tıklandığında **"Button1'e tıkladınız.",** button2'ye tıklandığında **"Button2'ye tıkladınız."** mesajlarını verdiriniz.
3. Oluşturduğunuz kodları aşağıdaki kutucuğa yazınız.

8. Uygulama

Mouse simgesi ile forma eklenen bir Button nesnesinin üzerine gelindiğinde **"Mouse şimdi üzerimdedir."** mesajını, mouse simgesi Button nesnesinin üzerinden kaldırıldığında **"Mouse artık üzerimde değildir."** mesajını verdiriniz.

```
private void button1_MouseMove(object sender, EventArgs e)
    {
        MessageBox.Show ("Mouse Üzerimdedir.");
    }
private void button1_MouseLeave(object sender, EventArgs e)
    {
        MessageBox.Show ("Mouse Artık Üzerimde Değildir.");
    }
```

Not

Bir nesne için sadece bir tane olay metodu oluşturulmaz. Nesne için tanımlanabilecek ne kadar olay varsa o kadar da olay metodu oluşturulabilir.

Sıra Sizde

1. Form üzerine 3 adet **Button** nesnesi ekleyiniz.
2. Form nesnesinin arka plan rengini turuncu yapınız.
3. Buttonların hepsinin arka plan ve yazı renklerini birbiriyle uyumlu olacak şekilde tasarlayınız.
4. • Mouse simgesi ile button1'in üzerine gelindiğinde **"Mouse benim üzerimdedir."**,
   • Mouse simgesi ile button2'nin üzerine çift tıklandığında **"Mouse iki kere tıklandı."**,
   • Button3'ün üzerinden mouse simgesi kaldırıldığında **"Mouse üzerinde değil."**
   mesajlarını verdiğiniz ve oluşturduğunuz kodları aşağıdaki kutucuğa yazınız.

Sıra Sizde

1. Bir Button nesnesine fare ile tıklandığında kaç farklı olay gerçekleşir? Araştırınız, gerçekleşen olayları not ediniz ve notlarınızı arkadaşlarınızla paylaşınız.
2. Bir TextBox nesnesine yazı yazıldığı andan itibaren kaç farklı olay gerçekleşir? Oluşturacağınız küçük gruplarla araştırınız, sonuçlarınızı diğer grupların sonuçlarıyla karşılaştırınız.

9. Uygulama

1. Adım: Form üzerine Görsel 1.24'te görüldüğü gibi GroupBox nesnesi içine 4 adet Button nesnesi ekleyiniz.

2. Adım: Form nesnesinin arka plan rengini beyaz yapınız.

3. Adım: Kırmızı yazılı Button nesnesine tıklandığında Form nesnesinin arka plan rengini kırmızı, yeşil yazılı Button nesnesine tıklandığında Form nesnesinin arka plan rengini yeşil, mavi yazılı Button nesnesine tıklandığında Form nesnesinin arka plan rengini mavi, gri yazılı Button nesnesine tıklandığında Form nesnesinin arka plan rengini gri yapan programı yazınız.

```
private void button1_Click(object sender, EventArgs e)
        {   this.BackColor = Color.Red;    }
private void button2_Click(object sender, EventArgs e)
        {   this.BackColor = Color.Green;  }
private void button3_Click(object sender, EventArgs e)
        {   this.BackColor = Color.Blue;   }
private void button4_Click(object sender, EventArgs e)
        {   this.BackColor = Color.Gray;   }
```

Görsel 1.24: Form boyama

10. Uygulama

1. Adım: Form üzerine Görsel 1.25'te görüldüğü gibi dört tane PictureBox nesnesini üst üste olacak şekilde yerleştiriniz.

2. Adım: PictureBox nesnelerinin "Visible" özelliğini "false" yapınız (Visible özelliği ile nesnenin görünürlüğü belirlenir.).

3. Adım: Dört tane RadioButton nesnesi ekleyiniz ve bunları numaralandırınız.

Görsel 1.25: Resim galerisi

radioButton1 nesnesi işaretli ise sadece pictureBox1 nesnesini görünür hâle getirecek, radioButton2 nesnesi işaretli ise sadece pictureBox2 nesnesini görünür hâle getirecek, radioButton3 nesnesi işaretli ise sadece pictureBox3 nesnesini görünür hâle getirecek, radioButton4 nesnesi işaretli ise sadece pictureBox4 nesnesini görünür hâle getirecektir.

```
private void radioButton1_CheckedChanged(object sender, EventArgs e)
        {  pictureBox1.Visible = radioButton1.Checked; }
private void radioButton2_CheckedChanged(object sender, EventArgs e)
        {  pictureBox2.Visible = radioButton2.Checked; }
private void radioButton3_CheckedChanged(object sender, EventArgs e)
        {  pictureBox3.Visible = radioButton3.Checked; }
private void radioButton4_CheckedChanged(object sender, EventArgs e)
        {  pictureBox4.Visible = radioButton4.Checked; }
```

Not

Bir nesne için sadece bir tane olay metodu oluşturulmaz. Nesne için tanımlanabilecek ne kadar olay varsa o kadar da olay metodu oluşturulabilir.

1.5. İSİM UZAYLARI (NAMESPACE)

İsim uzayları; program yazımı esnasında kullanılan metot, sınıf, değişken, sabit gibi yapıları mantıksal olarak kategorize etme sistemidir. İsim uzayları, proje büyüdükçe kod yapısının karmaşıklığını engeller. Örneğin aile fotoğraflarına bakmak isteyen biri sabit diskindeki dosyaların hepsini tek klasörde tutuyorsa aradığı fotoğrafa ulaşması çok zor ve karmaşık olacaktır. Dosyalarını müzik, video, fotoğraf klasörleri şeklinde düzenleyip hatta fotoğraflar klasörünün içine de aile, okul, doğa diye ayrı ayrı kategorilerde klasörler oluşturursa karmaşa ortadan kalkacaktır (Görsel 1.26).

Görsel 1.26: İsim uzayları

.NET Framework ile gelen standart isim uzayları kullanılabileceği gibi proje yazımı esnasında sonradan oluşturulan isim uzayları da kullanılabilir.

İsim uzayını projeye dâhil etmek için **using** kodu kullanılır. Varsayılan olarak **using** System; kod parçacığı projelerde bulunur. Bu kod parçacığındaki using ifadesi "kullanılıyor" anlamına gelir. System ifadesi ise kullanılan isim uzayını temsil eder.

```
using System.IO;
using System.Data;
using System.Windows.Forms;
```

Kodlara isim uzayları eklenebilir.

Not

Bu kütüphanelerin hepsi System içinde mevcuttur. Bu kütüphaneler eklendiğinde kodlar daha kısa yazılır. Örneğin "System.IO.File.WriteAllText" diye bir kod kullanılacaksa "using System.IO;" isim uzayı eklendiği için kodu sadece "File.WriteAllText" şeklinde yazmak yeterlidir.

Programda kişisel bir isim uzayı tanımlanarak kodların organize olması sağlanabilir.

Veri tabanına kayıt yapma işlemlerini içeren bir kütüphane geliştirilirken Kaydet isimli bir sınıf var ise bu kodlara ait isim uzayını tanımlamak için "namespace" anahtar sözcüğünün kullanımı yanda verilmiştir.

```
namespace VeriTabani {
    public class Kaydet
    {
     //kodlarınız
    }
}
```

Yukarıdaki Kaydet sınıfını kullanabilmek için programa using ifadesi ile isim uzayı eklenmelidir. Bu işlem yanda verilmiştir.

```
using VeriTabani
```

Sıra Sizde

İsim uzayı yazma ve projeye dâhil etme kurallarına dikkat ederek, kendi isim uzaylarınızı tanımlayıp projenize ekleyiniz.

1.6. DEĞİŞKENLER VE TEMEL VERİ TÜRLERİ

Değişkenler, programlamanın en temel kavramıdır. Program çalışması sırasında çeşitli türlerde verileri hafızada saklayan değişkenler, ihtiyaca göre tekrar tekrar kullanılan veri tutuculardır. Program çalıştığı sürece değişkenler RAM bellekte bulunur, program durdurulduğunda RAM bellekten silinir.

Görsel 1.27: İsim uzayları

Değişkenler; aynı anda içinde sadece tek ürün taşınabilen, farklı taşıma kapasitelerine sahip ve üzerlerinde seri numaraları yazılı **alışveriş sepetleri** gibi düşünülebilir (Görsel 1.27). Bu sepetlerden seçilen herhangi biri ile ihtiyaca göre ürün taşınabilir. Taşınacak ürünün boyutuna ve ağırlığına en uygun sepet seçimi yapılmalıdır. Büyük ebatta kutuya sahip bir ürün taşınacaksa büyük bir sepet seçilmelidir. Küçük ebatlı kutuya sahip bir ürün taşınacaksa taşıma işleminin en verimli şekilde yapılabilmesi için çok büyük bir sepet seçilmemelidir. Aynı seri numaralı sepetle gün içinde farklı ürünler de taşınabilir. Sepetin içeriği her taşımada sürekli değişebilir. Örneğin bir saat önce SP00052 seri numaralı sepete bakıldığında sepetin içinde parfüm olduğu görülürken on dakika önce muz, şimdi ekmek, sonrasında saatlerce sepetin boş bir şekilde kaldığı da görülebilir. Verilen örnekte sepetin boy ve ağırlık kapasitesi, **değişkenin veri türü**; "SP00052" ifadesi, **değişkenin ismi;** sepetin içindeki parfüm ise **değişkenin değeri** olarak düşünülebilir.

1.6.1. Temel Veri Türleri

Değişkenlerin içine aktarılacak verilerin türleri çeşitlilik gösterebilir. Her veri türünün RAM bellekte kapladığı alan ve değişkenin türüne göre verinin değer aralığı farklıdır.

Değişkenler, RAM bellekte yer kaplar ve programda kullanılan değişken sayısı arttıkça bu durum RAM belleğin kullanılabilir hafıza kapasitesini düşürür. Örneğin öğrenci notlarının girileceği bir değişkenin veri türü, değer aralığı 100 sayısına en yakın olan **byte** veya **sbyte** olarak tanımlanabilir. Değer aralığı,

100 sayısından çok daha fazla olan **short** ve **ushort** gibi veri türlerinde tanımlanmamalıdır. Bu, bir kişinin 100 odalı bir evde yaşamayı seçmesi gibi çok verimsiz, masraflı ve gereksiz bir eylemdir. Tablo 1.1, Tablo 1.2, Tablo 1.3 ve Tablo 1.4'te değişkenlerin veri türlerine göre kapladığı alan ve değer aralığı verilmiştir.

Tablo 1.1: Tam Sayı Veri Türleri

Tam Sayı Türü	Kapladığı Alan	Değer Aralığı
byte	1 Bayt	0,...,255
sbyte	1 Bayt	-128,...,127
short	2 Bayt	-32768,...,32767
ushort	2 Bayt	0,...65535
int	4 Bayt	-2147483648,...,2147483647
uint	4 Bayt	0,...,4294967295
long	8 Bayt	-9223372036854775808
ulong	8 Bayt	0,...,18446744073709551615

Tablo 1.2: Ondalık Sayı Veri Türleri

Ondalık Sayı Türü	Kapladığı Alan	Değer Aralığı
float	4 Bayt	±1.5*10-45,...,±3.4*1038
double	8 Bayt	±5.0*10-324,...,±1.7*10308
decimal	16 Bayt	±1.5*10-28,...,±7.9*1028

Tablo 1.3: Metinsel Veri Türleri

Metinsel Veri Türü	Kapladığı Alan	Değer Aralığı
string	Sınırsız	Metinsel ifade tutar.
char	2 Bayt	Tek bir karakter tutar.

Tablo 1.4: Mantıksal Veri Türleri

Mantıksal Veri Türü	Kapladığı Alan	Değer Aralığı
bool	1 bit	True-False veya 1-0

1.6.2. Değişken Tanımlama

(Değişkenin Veri Türü) (Değişen Adı)

```
int sayi
double pi
string soru
char karakter
bool cevap
```

Not

Aynı veri türündeki değişkenler, aynı kod satırında ve aralarına virgül konularak tanımlanabilir.

Örnek

```
int sayi,deger,sonuç;
string isim1,isim2,soyad;
```

Değişken isimleri belirli yazım standartlarına göre verilmelidir. Bu standartlara uymak zorunlu değildir fakat kodların okunabilirliği ve rahat anlaşılması açısından çok önemlidir. Programın tüm kodlarında aynı standarda göre değişken isimleri verilmelidir.

Programda en çok tercih edilen yazım standartları; camel case (deve gösterimi), snake case (yılan gösterimi), pascal case'dir.

a) Camel Case: Bu yazım standardına göre değişkenin ismindeki ilk kelimenin baş harfi küçük, diğer kelimelerin baş harfleri büyük olur.

Örnek string kullaniciAdi; int toplamHesapTutari;

b) Snake Case: Bu yazım standardına göre değişkenin ismindeki kelimelerin arasına alt çizgi (_) kullanılır.

Örnek string kullanici_adi; int toplam_hesap_tutari; string boy_Uzunlugu; string urun_Fiyati;

c) Pascal Case: Bu yazım standardına göre değişkenin ismindeki tüm kelimelerin baş harfleri büyük olur.

Örnek string KullaniciAdi; int ToplamHesapTutari; string BoyUzunlugu; string UrunFiyati;

1.6.3. Değişkene Değer Atama

(Değişkenin Adı) = (Değişkenin Değeri);

sayi = 52
PiSayisi = 3.14159
Soru = "Ülkemizde hangi deniz en kuzeydedir?"
karakter = 'A'
cevap=true
adi_soyadi = "Ali YOLCU";
sinav Not Ortalamasi = 76

Not

Değişken tanımlanırken de değer atama yapılabilir.

Örnek

int sayi=52;

1.6.4. Değişken İsimlendirme Kuralları

a) Değişken isimlerinde boşluk kullanılmaz. Boşluk yerine alt çizgi (_) kullanılabilir.

Örnek string ad soyad; şeklinde kullanım yanlıştır.
string ad_soyad; veya string adsoyad; şeklinde kullanım doğrudur.

b) ?, !, :, %, +, -, . gibi özel karakterler kullanılmaz.

Örnek string soru?; şeklinde kullanım yanlıştır.
string soru; şeklinde kullanım doğrudur.

c) Değişken isimleri sayı ile başlamaz.

Örnek byte 1not; şeklinde kullanım yanlıştır.
byte not1; şeklinde kullanım doğrudur.

ç) Değişken isimleri büyük ve küçük harfe duyarlıdır.

Örnek ulong toplamTutar; şeklinde tanımlanan değişken ile ulong toplamtutar; veya ulong ToplamTUTAR şeklinde tanımlanan değişken aynı değildir.

d) Herhangi bir kodla aynı isimde değişken tanımlanamaz. Değişkenlerde if, else, random gibi programa ait ifadeler isim olarak kullanılmaz.

e) Türkçe karakterlerin (ç, ö, ü, ğ, ş vb.) kullanılması tavsiye edilmez ancak kullanılmaması gibi bir zorunluluk da yoktur.

Sıra Sizde

Aşağıdaki değişken tanımlamalarını doğru ya da yanlış olarak değerlendiriniz. Yanlış olan değişken tanımlamalarının sebebini açıklamalar bölümüne yazınız.

Değişken Adı	Doğru	Yanlış	Açıklamalar
Musteri_Tc			
Müsteri_adı			
1.isim			
toplam			
textBox			
Yüzde%18			
seri no			
Yuzde10			

1.6.5. Değişken Veri Türü Dönüştürme (Convert) İşlemleri

Değişkenlerin veri türlerini bazen değiştirmek gerekebilir. Sayısal bir ifade, bir nesnenin Text özelliğine aktarılmak istendiğinde program hata verecektir. İçeriği tamamen sayı olsa da metinsel bir ifadeyi sayısal veri türüne sahip bir değişkene aktarırken program yine hata verecektir. Bu tip durumlarda değişkenlerin veri türlerini dönüştürmek gerekir. Tür dönüşümünü sağlayacak hazır metotlar şunlardır:

ToString() >>>> Her türden değişkeni string türüne dönüştürür. ToString(), en sık kullanılan dönüştürme metodudur.

```
Convert.ToByte(metin)      >>>>> Byte'a çevirir.
Convert.ToInt16(metin)     >>>>> Short'a çevirir.
Convert.ToInt32(metin)     >>>>> Int'e çevirir.
Convert.ToInt64(metin)     >>>>> Long'a çevirir.
Convert.ToSingle(metin)    >>>>> Float'a çevirir.
Convert.ToDouble(metin)    >>>>> Double'a çevirir.
Convert.ToDecimal(metin)   >>>>> Decimal'a çevirir.
Convert.ToChar(metin)      >>>>> Char'a çevirir.
Convert.ToBoolean(metin)   >>>>> Bool'a çevirir.
```

Örnek

```
int sayi1=100;
string deger;
deger=sayi1; //Program bu noktada convert type 'int' to 'string' şeklinde hata verecektir.
```

Örnek

```
int sayi1=100;
 string deger;
 deger=sayi1.ToString(); //Program artık hata vermeyecektir çünkü veri türleri uyumludur.
```

11. Uygulama

1. Adım: Form üzerine Görsel 1.28'de görüldüğü gibi bir CheckBox nesnesi ekleyiniz.

2. Adım: İki adet Label nesnesi ekleyiniz.

3. Adım: CheckBox nesnesi işaretli ise label2 nesnesine "True" değerini, CheckBox nesnesi işaretli değil ise label2 nesnesine "False" değerini gösteren programı yazınız.

Görsel 1.28: Lamba kontrol

```
private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {
            bool secim;
            secim = checkBox1.Checked;  //Checked özelliği True veya False değerleri alır.
            label2.Text = secim.ToString();
        }
```

Sıra Sizde

Bir değişken tanımlayıp içine 500 sayı değerini atayınız. Butona tıklandığında bu değişkendeki değeri ekrana mesaj olarak verdiriniz.

1.7. ARİTMETİKSEL OPERATÖRLER

Aritmetiksel operatörler, matematiksel işlemlerde kullanılan özel karakterlerdir (Tablo 1.5).

Tablo 1.5: Aritmetiksel Operatörler

Operatör Adı	Sembolü	Örnek
Toplama	+	2+8
Çıkarma	-	8-2
Çarpma	*	8*2
Bölme	/	8/2
Mod alma	%	8%2

Not Bir sayı başka bir sayıya bölündüğünde kalan sayıya **mod** denir.

Örnek : 8%2=0 iken 8%5=3 sonucunu verir.

12. Uygulama

Girilen iki sayıyı toplayan programı Görsel 1.29'daki gibi tasarlayıp yazınız.

```
private void button1_Click(object sender, EventArgs e)  {
        int sayi1, sayi2, toplam;
        sayi1=Convert.ToInt16(textBox1.Text);
        sayi2 = Convert.ToInt16(textBox2.Text);
        toplam = sayi1 + sayi2;
        textBox3.Text = toplam.ToString();
```

Görsel 1.29: Lamba kontrol

Not :

TextBox nesnesinin içine sayısal bir değer girilmiş olsa bile matematiksel işlemler yapılamaz çünkü TextBox nesnesi metinsel veri türüne sahip değerler alabilen bir nesnedir. Bu tip durumlarda Convert kodu ile veri türü dönüşümü yapılmalıdır. Ayrıca TextBox nesnesine sayısal veri türünde bir değer yazmak için ToString() metodu ile veri türü dönüşümü yapılmalıdır.

13. Uygulama

Aşağıdaki örnek kodları çalıştırınız.

Birinci Örnek Kodlar	İkinci Örnek Kodlar
string ad, soyad,topla; ad = "Zeynep"; soyad = "Sare"; topla = ad +" "+ soyad; MessageBox.Show(topla);	textBox1.Text="25"; textBox2.Text="2" textBox3.Text= textBox1.Text + textBox2.Text

Not

Programlama dillerinde toplama operatörü sadece sayıları toplamak için kullanılmaz. Metinsel veri türüne sahip değişkenlerin arasına artı (+) operatörü konularak string değerlerini birleştirmek için de toplama operatörü kullanılır. Birinci örnek kodlar çalıştığında ekrana "Zeynep Sare" mesajı gelir. İkinci örnek kodlar çalıştığında ise textBox3'ün değeri "252" olur.

Sıra Sizde

1. Girilen iki sayının toplama, çıkarma, çarpma ve bölme işlemlerini yapan programı tasarlayıp yazınız.
2. Button nesnesine çift tıklandığında TextBox nesnesine girilen sayının karesini alıp mesaj verdiren programı, Görsel 1.30'da görüldüğü gibi tasarlayıp yazınız.

Görsel 1.30: Kare alma uygulaması

1.8. İŞLEM ÖNCELİĞİ

Matematiksel bir ifade yazılan kod satırında birden fazla aritmetiksel operatör kullanılabilir. Böyle durumlarda sonuçların doğru çıkması için operatörler arasındaki işlem önceliklerine dikkat edilerek kod yazılmalıdır.

Kodlar çalıştırılırken öncelikle varsa parantez içindeki işlemler, daha sonra parantez dışındaki işlemler yapılır. Çarpma, bölme ve mod alma işlemleri kendi aralarında; toplama ve çıkarma işlemleri de kendi aralarında aynı önceliğe sahiptir. Kendi aralarında aynı önceliğe sahip operatörlerin olduğu ifadelerde ise sırasıyla soldan sağa doğru işlem yapılır.

Örnek : 4x3+15/3-1 işlemini yapınız.

Çözüm:

=12+5-1 -->Önce çarpma sonra bölme işlemi yapılır.

=17-1 -->Toplama işlemi ve sonrasında çıkarma işlemi yapılır.

=16

Örnek : (3+4-2)x(10/5x2)-4 işlemini yapınız.

Çözüm:

=5x4-4 -->Önce parantez içindeki işlemler yapılır.

=20-4 -->Çarpma işlemi, toplama işleminden öncelikli olarak yapılır.

=16

Not

Kod yazımı sırasında karmaşayı ortadan kaldırmak ve en doğru sonuca ulaşmak için öncelikli olan ifadeler parantez içine alınır.

Sıra Sizde

1. Siz de ikili gruplar hâlinde içinde aritmetiksel operatörler geçen 10 adet ifade yazınız ve kendi sonuçlarınızla grup arkadaşınızın sonuçlarını karşılaştırınız.

2. Aşağıdaki işlemlerin sonucunu yazınız.

7x5-3		(5x4)-(3+9)	
20+30-4x10		(5x8/4)+3-5x2	
9/3+2x2-5+1			

14. Uygulama

İki sayıyı toplayıp mesaj verdiren programı yazınız (Değişkenlere istenilen değerler atanabilir.).

```
private void button1_Click(object sender, EventArgs e)
{   int sayi1;
    int sayi2;
    int toplam;
    sayi1 = 5;
    sayi2 = 20;
    toplam = sayi1 + sayi2;
    MessageBox.Show(toplam.ToString());
}
```

Sıra Sizde

İki sayıyı toplayıp, 2 ile çarparak, çıkan sonuca 5 ekleyen ve oluşan değeri mesaj verdiren programı işlem önceliklerine dikkat ederek yazınız (Değişkenlere istenilen değerler atanabilir.).

```
private void button1_Click(object sender, EventArgs e)
{



}
```

15. Uygulama

http://kitap.eba.gov.tr/KodSor.php?KOD=21071

1. Adım: Form üzerine Görsel 1.31'de görüldüğü gibi etiket fiyatı girişi için TextBox nesnesi ekleyiniz.

2. Adım: Dört adet Button nesnesi ekleyiniz.

3. Adım: Button nesnelerinin arka plan renklerini değiştiriniz.

Görsel 1.31: İndirimli ürün

4. Adım: İndirim oranlarını Button nesnelerinin üzerine yazınız.

5. Adım: TextBox nesnesine etiket fiyatı girilen ürünün, seçilen indirim oranına göre indirimli fiyatını bulup Label nesnesinde gösteren programı yazınız.

```
private void button1_Click(object sender, EventArgs e)
    {
        int etiketFiyati;
        double indirimliFiyat;
        etiketFiyati = Convert.ToInt32(textBox1.Text);
        indirimliFiyat = etiketFiyati-etiketFiyati * 0.10; //Yüzde 10 indirim
        label3.Text = indirimliFiyat.ToString();
    }
```

```
private void button2_Click(object sender, EventArgs e)
    {
        int etiketFiyati;
        double indirimliFiyat;
        etiketFiyati = Convert.ToInt32(textBox1.Text);
        indirimliFiyat = etiketFiyati-etiketFiyati * 0.25; //Yüzde 25 indirim
        label3.Text = indirimliFiyat.ToString();
    }
```

```
private void button3_Click(object sender, EventArgs e)
    {
        int etiketFiyati;
        double indirimliFiyat;
        etiketFiyati = Convert.ToInt32(textBox1.Text);
        indirimliFiyat = etiketFiyati-etiketFiyati * 0.50; //Yüzde 50 indirim
        label3.Text = indirimliFiyat.ToString();
    }
```

```
private void button4_Click(object sender, EventArgs e)
    {
        int etiketFiyati;
        double indirimliFiyat;
        etiketFiyati = Convert.ToInt32(textBox1.Text);
        indirimliFiyat = etiketFiyati-etiketFiyati * 0.75; //Yüzde 75 indirim
        label3.Text = indirimliFiyat.ToString();
    }
```

Not

"etiketFiyati - etiketFiyati x 0.10" ifadesinde işlem önceliği çarpmada olduğu için önce çarpma işlemi sonra çıkarma işlemi yapılacaktır. Aynı ifade "etiketFiyati - (etiketFiyati x 0.10)" şeklinde yazılarak da işlem yapılabilir.

16. Uygulama

http://kitap.eba.gov.tr/KodSor.php?KOD=21072

1. Adım: Form üzerine Görsel 1.32'de görüldüğü gibi 1 adet GroupBox, 1 adet Button, 5 adet Label, 6 adet TextBox nesnesi ekleyiniz.

2. Adım: Türkçe ve matematik netlerinin gösterileceği TextBox nesnelerinin Enabled özelliğini bilgi girişini engellemek için false yapınız.

Görsel 1.32: Net hesaplama

3. Adım: Button nesnesi tıklandığında ilgili derslere ait net sayısını hesaplayan programı yazınız (Net sayısı, doğru sayısından yanlış sayısının dörtte biri çıkarılarak hesaplanır.).

```
private void button1_Click(object sender, EventArgs e)
    {
        double turkceDogru, matDogru;
        double turkceYanlis, matYanlis;
        double turkceNet, matNet;
        turkceDogru = Convert.ToDouble(textBox1.Text);
        turkceYanlis = Convert.ToDouble(textBox2.Text);
        turkceNet = (turkceDogru - (turkceYanlis / 4));
        textBox3.Text = turkceNet.ToString();
        matDogru = Convert.ToDouble(textBox4.Text);
        matYanlis = Convert.ToDouble(textBox5.Text);
        matNet = (matDogru - (matYanlis / 4));
        textBox6.Text = matNet.ToString();
    }
```

ÖLÇME VE DEĞERLENDİRME

A) Aşağıdaki cümlelerde parantez içine yargılar doğru ise "D", yanlış ise "Y" yazınız.

1. ( ) Kod editörü platformu kullanılarak sadece C# programlama dilinde kodlama yapılır.

2. ( ) C# programlama dilinde, string veri türüne sahip bir değişken int veri türüne dönüştürebilme özelliğine sahiptir.

3. ( ) int a="52"; hatalı bir koddur.

4. ( ) string a="16"; hatalı bir koddur.

5. ( ) Değişken isimleri, sayı ile başlayabilme özelliğine sahiptir.

6. ( ) Bir nesne için aynı anda birden fazla olay metodu oluşturma özelliği bulunur.

7. ( ) Button nesnesi için sadece Click olay metodu oluşturulur.

B) Aşağıda verilen cümlelerdeki boşlukları kutularda verilen ifadelerle tamamlayınız.

form	ToString()	%	ToolBox	using

8. İsim uzayını projeye dâhil etmek için ..... kodu kullanılır.

9. Form üzerine eklenen nesneler ..... panelinden seçilir.

10. Mod alma işlemi için ..... aritmetiksel operatörü kullanılır.

11. Sayısal veri türüne sahip bir değişken ..... metodu ile string veri türüne dönüştürülür.

C) Aşağıdaki kod blokunda boş bırakılan yerlere doğru sözcükleri yazınız.

```
using .....;
.....  ResimEkle {
    public class Kaydet
   {
    //kodlarınız
   }
}
```

12. Aşağıdaki kod satırlarından hangisi çalışmaz? Kodun çalışmama sebebini yazınız.

int öğrenci no;	
int 1.not;	
string isim;	
double yaricap;	
bool cinsiyet;	
int sayi1=13.4;	
char karakter = "kmr";	
string tc = "00334412376";	

13. Aşağıdaki kodlar çalıştırıldığında ekranda görülecek sayı kaçtır?

```
int öğrenci no;
int 1.not;
string isim;
double yaricap;
bool cinsiyet;
int sayi1=13.4;
char karakter = “kmr”;
string tc = “00334412376”;
```

14. TextBox nesnesine girilen sayının %18’ini bulup mesaj verdiren programı yazınız.

15. Yarıçapı girilen dairenin alanını ve çevresini hesaplayan programı yazınız.

2. ÖĞRENME BİRİMİ

KARAR VE DÖNGÜ YAPILARI

KONULAR

2.1. KARAR İFADELERİ

2.2. MANTIKSAL OPERATÖRLER

2.3. DÖNGÜLER

2.4. HATA AYIKLAMA

ANAHTAR KELİMELER

Karar ifadeleri, döngü, if, else, for, while, hata ayıklama

NELER ÖĞRENECEKSİNİZ?

- Karşılaştırma operatörleri
- Karar ifadelerinin kullanımı
- if, if-else, else if ifadelerinin kullanımı
- İç içe karar ifadelerinin kullanımı
- Mantıksal operatörler
- Döngü çalışma mantığı
- For, while, do-while döngülerinin kullanımı

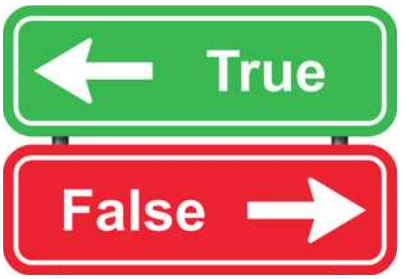
HAZIRLIK ÇALIŞMALARI

1. Günlük yaşantınızda şarta bağlı olarak yaptığınız işlemlerin neler olduğunu arkadaşlarınızla tartışınız.
2. Program yazarken döngü yapılarının sağladığı kolaylıklar neler olabilir? Araştırınız.

2.1. KARAR İFADELERİ

Karar ifadeleri, tüm programlama dillerinde bulunan, koşula veya koşullara bağlı bir şekilde programın nasıl ilerleyeceğini belirleyen yapılardır. Programın yol ayrımı noktalarını karar ifadeleri temsil eder. Bu yapı sayesinde programlar daha dinamik ve etkileşimlidir.

Karar ifadeleri, programlamanın vazgeçilmez yapılarıdır. Örneğin profil girişi için gerekli olan **kullanıcı adı ve şifre** bilgileri, karar ifadeleri kullanılarak kontrol edilir ve program tarafından girişe izin verilir veya verilmez. Bu durum, programın ilerleyeceği yönü belirler (Görsel 2.1). TextBox nesnesine girilebilecek en büyük sayının 100 olması istenirse karar ifadeleri ile TextBox nesnesinin değerinin kontrol edilmesi ve veri girişinin sağlanması da bu konuya bir başka örnektir.



Görsel 2.1: Karar ifadeleri

2.1.1. Karşılaştırma Operatörleri

Karşılaştırma operatörleri, karar ifadelerinde iki değeri birbiriyle karşılaştırmak için kullanılan operatörlerdir (Tablo 2.1).

Tablo 2.1: Karşılaştırma Operatörleri

Operatör	Anlamı
<	Küçükse
>	Büyükse
==	Eşitse
<=	Küçük veya Eşitse
>=	Büyük veya Eşitse
!=	Eşit Değilse

2.1.2. if Yapısı

if, kelime olarak eğer anlamına gelir. Sadece şart sağlandığında çalışması istenen kodlar için kullanılır. Şart ifadesi sağlandığında true, sağlanmadığında false değeri oluşur. if(true) olduğunda if yapısına bağlı kodlar çalışır, if(false) olduğunda kodlar çalışmaz.

```
if(şart ifadesi)
        {
          // Şart ifadesi sağlanıyorsa çalışacak kodlar
        }
```

1. Uygulama

1. Adım: Form üzerine Button nesnesi ekleyiniz.

2. Adım: Button nesnesinin Click olay metoduna aşağıdaki kodları yazınız.

```
byte skor1,skor2;
        skor1 = 4;
        skor2 = 1;
        if (skor1 > skor2)
        {
            MessageBox.Show("1. Takım Kazandı.");
        }
```

Not Birinci uygulamada **(skor1 > skor2)** şart ifadesidir. Şart ifadesi sağlandığında çalışması istenen kodlar küme parantezlerinin içine yazılır. Şart sağlandığı için parantez içindeki kod çalışacaktır. Şart sağlanmadığında parantez içindeki kod çalışmayacak, parantezden sonra hangi kod varsa onlar işletilecektir.

2. Uygulama

1. Adım: Form üzerine Button ve TextBox nesnesi ekleyiniz.

2. Adım: Button nesnesi tıklandığında TextBox nesnesine girilen sayı 17'den büyük ise "Ehliyet başvurusunda bulunabilirsiniz." şeklinde mesaj veren programı yazınız.

```
private void button1_Click(object sender, EventArgs e)
    {
        byte yas;
        yas = Convert.ToByte(textBox1.Text);
        if (yas > 17)
        {
            MessageBox.Show("Ehliyet başvurusunda bulunabilirsiniz.");
        }
    }
```

Not Yukarıdaki kod blokunda şart ifadesinde kullanılan "yas" değişkeni yerine "Convert.ToByte(textBox1.Text)" ifadesi de kullanılabilir.

3. Uygulama

1. Adım: Form üzerine 2 adet Label, 2 adet TextBox ve 1 adet Button nesnesi ekleyiniz.

2. Adım: Programı Görsel 2.2'de görüldüğü gibi tasarlayınız.

3. Adım: Button nesnesi tıklandığında textBox1'deki değer ile textBox2'deki değeri büyüklük, küçüklük ve eşitlik bakımından karşılaştıran ve sonucu ekrana mesaj olarak veren programı yazınız.

Görsel 2.2: Sayıları karşılaştırma uygulaması

```
private void button1_Click(object sender, EventArgs e)
{
    byte sayi1, sayi2;
    sayi1 = Convert.ToByte(textBox1.Text);
    sayi2 = Convert.ToByte(textBox2.Text);
    if (sayi1 > sayi2)
    {
    MessageBox.Show("1.sayı 2.sayıdan büyüktür.");
    }
    if (sayi1 == sayi2)
   {
   MessageBox.Show("Sayılar birbirine eşittir.");
   }
    if (sayi1 < sayi2)
    {
   MessageBox.Show("2.sayı 1.sayıdan büyüktür.");
    }
 }
```

4. Uygulama

1. Adım: Form üzerine Button ve TextBox nesnesi ekleyiniz.

2. Adım: Button nesnesi içine tıklandığında TextBox nesnesine girilen sayının tek mi, çift mi olduğunu bulan programı yazınız.

Not Bir sayının 2'ye bölümünden kalan sayı 0 ise bölünen sayı çifttir, kalan sayı 1 ise bölünen sayı tektir.

```
int sayi;
sayi =Convert.ToInt32(textBox1.Text);
if(sayi % 2 == 0)
{
MessageBox.Show("Bu bir çift sayıdır.");
}
if(sayi % 2 ==1)
{
MessageBox.Show("Bu bir tek sayıdır.");
}
```

2.1.3. if-else Yapısı

else; kelime olarak **değilse, aksi durumda** anlamına gelir. Şart ifadesi sağlandığında if kod bloku içindeki kodlar çalışır, şart ifadesi sağlanmadığında ise else kod bloku içindeki kodlar çalışır.

```
if(şart ifadesi)
    {
      // Şart ifadesi sağlanıyorsa çalışacak kodlar
    }
else
    {
      // Şart ifadesi sağlanmadığında çalışacak kodlar
    }
```

5. Uygulama

1. Adım: Form üzerine TextBox ve Button nesnesi ekleyiniz.

2. Adım: TextBox nesnesine girilen kullanıcı adı milliegitim@meb.k12.tr'ye eşit ise "Kullanıcı sisteme kayıtlıdır." mesajını, kullanıcı adı milliegitim@meb.k12.tr'ye eşit değil ise "Kullanıcı adınız yanlıştır." mesajını veren programı yazınız.

```
string kullanici_adi;
kullanici_adi=textBox1.Text;
if (kullanici_adi == "milliegitim@meb.k12.tr")
{
MessageBox.Show("Kullanıcı sisteme kayıtlıdır.");
}
else
{
MessageBox.Show("Kullanıcı adınız yanlıştır.");
}
```

Not Beşinci uygulamadaki kod blokunda (**kullanici_adi == "milliegitim@meb.k12.tr"**) şart ifadesi sağlanmadığında **else** kod blokundaki kodlar çalışır.

Sıra Sizde

TextBox nesnesine girilen sayının tek mi, çift mi olduğunu bulan programı if-else yapısını kullanarak yazınız.

6. Uygulama

1. Adım: Form üzerine 2 adet GroupBox, 2 adet CheckBox, 1 adet Button ve 1 adet ListBox nesnesi ekleyiniz.

2. Adım: Form ve GroupBox nesnelerinin renklerini beyaz yapınız.

3. Adım: Button nesnesine tıklandığında "Lamba Aç/Kapa" işaretli ise "Lambalar Açık" mesajını, "Lamba Aç/Kapa" işaretli değil ise "Lambalar Kapalı" mesajını, "Kombi Aç/Kapa" işaretli ise "Kombi Açık" mesajını, "Kombi Aç/Kapa" işaretli değil ise "Kombi Kapalı" mesajını ListBox nesnesine ekleyen programı Görsel 2.3'teki gibi tasarlayıp yazınız.

Görsel 2.3: Akıllı ev

```
if (checkBox1.Checked == true)
 {
  listBox1.Items.Add("Lambalar Açık");
  }
 else
 {
 listBox1.Items.Add("Lambalar Kapalı");
  }
 if (checkBox2.Checked == true)
 {
  listBox1.Items.Add("Kombi Açık");
 }
else
   {
   lis tBox1.Items.Add("Kombi Kapalı");
   }
```

7. Uygulama

1. Adım: Form üzerine 2 adet RadioButton, 2 adet TextBox, 4 adet Label, 1 adet Button ve 1 adet GroupBox nesnesi ekleyiniz.

2. Adım: Programı Görsel 2.4'te görüldüğü gibi tasarlayınız.

3. Adım: Button nesnesine tıklanıp RadioButton nesnelerinden "Toplama İşlemi" işaretlendiğinde sayıları toplayan, "Çarpma İşlemi" işaretlendiğinde sayıları çarpan ve sonucu ekrandaki Label nesnesinde gösteren programı if-else yapısını kullanarak yazınız.

Görsel 2.4: Toplama / Çarpma uygulaması

```
private void button1_Click(object sender, EventArgs e)
 {
     int deger1 = Convert.ToInt32(textBox1.Text);
     int deger2 = Convert.ToInt32(textBox2.Text);
     int sonuc;
      if (radioButton1.Checked == true)
      {
       label4.Text = (deger1 + deger2).ToString();
        }
       else
       {
       label4.Text = (deger1 - deger2).ToString();
       }
 }
```

2.1.4. else if Yapısı

Şartın sağlanmadığı durumlarda else if kullanılarak yeni bir şart ifadesi daha yazılabilir. else if ifadesinden sonra tekrar else if ifadesi veya else ifadesi kullanılabilir.

```
if(şart ifadesi)
  {
   // if şart ifadesi sağlanıyorsa çalışacak kodlar
   }
 else if(şart ifadesi)
  {
  // else if şart ifadesi sağlanıyorsa çalışacak kodlar
   }
```

8. Uygulama

http://kitap.eba.gov.tr/KodSor.php?KOD=21073

1. Adım: Form üzerine TextBox ve Button nesnesi ekleyiniz.

2. Adım: Button nesnesine tıklandığında TextBox nesnesinin içine yüzlük sistemde girilen bir notu beşlik sisteme çeviren programı yazınız.

```
int sayi;
 sayi = Convert.ToInt32(textBox1.Text);
 if (sayi < 0)
 { MessageBox.Show("0'dan büyük bir sayı giriniz!"); }
 else if (sayi < 25)
  {
  MessageBox.Show("Notunuz 0");
  }
  else if (sayi < 45)
  {
 MessageBox.Show("Notunuz 1");
  }
  else if (sayi < 55)
  {
  MessageBox.Show("Notunuz 2");
  }
  else if (sayi < 70)
  {
  MessageBox.Show("Notunuz 3");
}
 else if (sayi < 85)
 {
 MessageBox.Show("Notunuz 4");
 }
else if (sayi <= 100)
{
 MessageBox.Show("Notunuz 5");
}
 else
{
 MessageBox.Show("Hatalı giriş yaptınız!");
  }
```

Sıra Sizde

Hava sıcaklığı 10 derecenin altında ise "Hava soğuk" mesajını, hava sıcaklığı 10-25 derece arasında ise "Hava hafif sıcak" mesajını, hava sıcaklığı 25 derecenin üstünde ise "Hava sıcak" mesajını veren programı yazınız.

2.1.5. İç İçe Şart İfadeleri

İç içe şart ifadeleri, birbirini izleyen birden çok şart ifadesinin kontrolünü gerçekleştirmek için kullanılır.

```
if(şart ifadesi)
{
    if(şart ifadesi)
    {
      if(şart ifadesi)
        {

         }
         else if(şart ifadesi)
        {

        }
    }
}
```

Not : Programın mantığına göre şart ifadeleri istenildiği kadar birbiri içinde yazılabilir. İç içe şart ifadeleri kullanılırken karmaşık bir kod yazımı olmaması için küme parantezlerini kendi aralarında hizalamaya dikkat edilmelidir.

9. Uygulama

1. Adım: Form nesnesi üzerine 2 adet Label, 1 adet TextBox ve 1 adet Button nesnesi ekleyiniz.

2. Adım: Programı Görsel 2.5'teki gibi tasarlayınız.

3. Adım: Kontrol et butonuna tıklandığında not ortalaması girilen kişinin notu 85 veya üzeri ise "Takdir Belgesi Almaya Hak Kazandınız." mesajını, not ortalaması girilen kişinin notu 70 veya üzeri ise "Teşekkür Belgesi Almaya Hak Kazandınız." mesajını, not ortalaması girilen kişinin notu 50'nin altında ise "Sınıf Geçmek İçin Yeterli Not Alamadınız." mesajını veren programı yazınız.

Görsel 2.5: Toplama / Çarpma uygulaması

```
private void button1_Click(object sender, EventArgs e)
 {
      byte ortalama;
      ortalama = Convert.ToByte(textBox1.Text);
      if (ortalama >= 50)
      {
      if (ortalama >= 85)
      {
       label1.Text = "Takdir Belgesi Almaya Hak Kazandınız.";
       }
       else if (ortalama >= 70)
       {
       label1.Text = "Teşekkür Belgesi Almaya Hak Kazandınız.";
       }
       else
      { label1.Text = "Belge Almadan Sınıf Geçtiniz.";   }
      }
      else
     { label1.Text = "Sınıf Geçmek İçin Yeterli Not Alamadınız."; }
   }
```

2.1.6. Switch-Case

Switch-case, bir ifadenin aldığı değere bağlı olarak program için birçok farklı çalışma yolu belirleyen bir komuttur. Switch ifadesindeki değer hangi durumun değeri ile eşleşiyorsa o duruma ait kodlar çalışır. Hiçbir durum ile eşleşme olmaz ise default ifadesinde belirtilen kodlar çalışır. "break;" komutu, kodlar çalıştıktan sonra switch-case ifadesinden çıkmayı sağlar. Switch ifadesi içine yazılan değerin veri türü ile case ifadelerindeki değerin veri türleri aynı olmalıdır. Case ifadelerindeki değerler için değişken kullanılmaz.

```
switch (sayısal değer){
   case 1:
      //kodlar
      break;
   case 2:
      //kodlar
      break;
   case 3:
      //kodlar
      break;
   default:
      //kodlar
      break;
}
```

```
switch (karakter değer){
   case 'X':
      //kodlar
      break;
   case 'Y':
      //kodlar
      break;
   case 'Z':
      //kodlar
      break;
   default:
      //kodlar
      break;
}
```

```
switch (metinsel değer){
   case "kırmızı":
      //kodlar
      break;
   case "yeşil":
      //kodlar
      break;
   case "mavi":
      //kodlar
      break;
   default:
      //kodlar
      break;
}
```

Switch-case ile yapılabilecek tüm işlemler if, if-else, else if yapıları kullanılarak da yapılabilir. Kod karmaşıklığını ortadan kaldırmak için uygun durumlarda switch-case kullanılabilir.

10. Uygulama

1. Adım: Form üzerine Button nesnesi ekleyiniz.

2. Adım: Button nesnesine tıklandığında bilgisayarın tarih bilgisine göre haftanın hangi gününde olduğunuzu bulan programı yazınız.

Not

DateTime.Now.DayOfWeek kodu ile haftanın kaçıncı gününde olunduğu bilgisi elde edilir.

```
int gun = Convert.ToInt32(DateTime.Now.DayOfWeek);
switch (gun)
{
case 1:
MessageBox.Show ("Pazartesi");
break;
case 2:
MessageBox.Show ("Salı");
break;
case 3:
MessageBox.Show ("Çarşamba");
break;
case 4:
MessageBox.Show ("Perşembe");
break;
case 5:
MessageBox.Show ("Cuma");
break;
case 6:
MessageBox.Show ("Cumartesi");
break;
case 0:
MessageBox. Show ("Pazar");
break;
default:
MessageBox.Show("Hata oluştu.");
break;
}
```

11. Uygulama

1. Adım: Form üzerine Button nesnesi ekleyiniz.

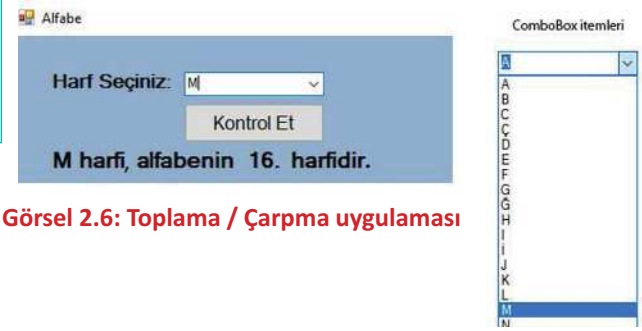
2. Adım: Button nesnesine tıklandığında bilgisayarın tarih bilgisine göre o günün hafta içi mi, hafta sonu mu olduğunu bulan programı yazınız.

Not On birinci uygulamada görüldüğü gibi birden fazla durum için aynı kodlar çalıştırılabilir.

```
int gun = Convert.ToInt32(DateTime.Now.DayOfWeek);
switch (gun)
{
case 1:
case 2:
case 3:
case 4:
case 5:
MessageBox.Show("Hafta içi");
break;
case 6:
case 0:
MessageBox.Show("Hafta sonu");
break;
default:
MessageBox.Show("Hata oluştu.");
break;
}
```

Sıra Sizde

1. ComboBox nesnesi içinden seçilen bir harfin, alfabenin kaçıncı harfi olduğunu bulan programı switch-case kullanarak, Görsel 2.6'da görüldüğü gibi tasarlayıp yazınız.
2. ComboBox nesnesi içinden seçilen bir harfin sesli harf mi, sessiz harf mi olduğunu bulan programı yazınız.



Görsel 2.6: Toplama / Çarpma uygulaması

2.2. MANTIKSAL OPERATÖRLER

Kod blokları, if yapılarında mantıksal operatörler kullanılmadığı zaman tek bir şart ifadesinin sonucuna bağlı olarak çalışır. Bazen birden fazla şart ifadesinin kullanılması gerekebilir. Örneğin kullanıcı doğrulama işlemi için kullanıcı adının ve şifre bilgilerinin aynı şart ifadesinde kontrol edilmesi gerekir (Görsel 2.7). Bu durumda Tablo 2.2'deki mantıksal operatörlere ihtiyaç vardır.

Görsel 2.7: Mantıksal operatörler

Tablo 2.2: Mantıksal Operatörler

Operatör Adı	Sembolü	Örnek
AND (ve)	&&	(A<B) && (A<C)
OR (veya)	||	(A<B) || (A<C)
NOT (değil)	!	!(A<B)
ulong	8 Bayt	0,...,18446744073709551615

Not İç içe if yapıları kullanılarak birden fazla şart ifadesi kontrol edilebilir fakat bu durum, kod yapısını karmaşık hâle getirebilir. Örneğin dört tane şart ifadesi için iç içe if yapısı kullanılırsa kodu okumak ve yazmak zorlaşır.

2.2.1. AND (&&) Operatörü

Şart ifadelerinin hepsinin sağlanması gerektiğinde "and" operatörü kullanılır. Şart ifadelerinden herhangi biri sağlanmadığında if yapısına bağlı kod bloku çalışmaz.

```
if(user_name=="admin@meb.k12.tr" && password == "12345")
{
// Kodlarınız
}
if(user_name=="admin@meb.k12.tr" && password == "12345" && yas >= 15)
 {
  // Kodlarınız
 }
```

Not Yeni bir && operatörü kullanılarak şart ifadelerinin sayısı artırılabilir.

12. Uygulama

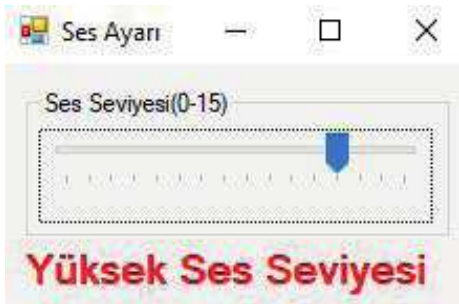
http://kitap.eba.gov.tr/KodSor.php?KOD=21074

1. Adım: Form üzerine 1 adet GroupBox, 1 adet TrackBar, 1 adet Label nesnesi ekleyiniz.

2. Adım: TrackBar nesnesi için "Scroll" olay metodu oluşturunuz (TrackBar nesnesi kaydırıldığında Scroll olayı tetiklenir.).

3. Adım: TrackBar nesnesinin Minimum özelliğini "0", Maximum özelliğini "15" yapınız.

4. Adım: TrackBar nesnesinin Value özelliği 0 ise siyah renkli yazı ile Label nesnesine "Ses Yok", TrackBar nesnesinin Value özelliği 1 ile 10 arasında ise yeşil renkli yazı ile Label nesnesine "Normal Ses Seviyesi", TrackBar nesnesinin Value özelliği 11 ile 15 arasında ise kırmızı renkli yazı ile Label nesnesine "Yüksek Ses Seviyesi" mesajlarını veren programı Görsel 2.8'deki gibi tasarlayıp yazınız (Value özelliği, TrackBar nesnesinin hangi değere sahip olduğu bilgisini verir.).



Görsel 2.8: Ses ayarı

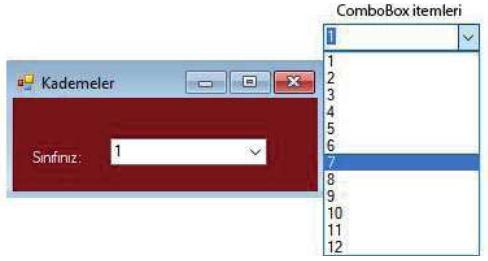
```
private void trackBar1_Scroll(object sender, EventArgs e)
 {  int ses = trackBar1.Value;
        if (ses == 0)
        {
        label1.Text = "Ses Yok";      label1.ForeColor = Color.Black;
         }
        if (ses > 0 && ses <= 10)
        {
        label1.Text = "Normal Ses Seviyesi";     label1.ForeColor = Color.Green;
         }
        if (ses >= 11)
        {
        label1.Text = "Yüksek Ses Seviyesi";      label1.ForeColor = Color.Red;
        }
 }
```

13. Uygulama

1. Adım: Form üzerine Label ve ComboBox nesnesi ekleyiniz.

2. Adım: ComboBox nesnesinin içine 1 ile 12 arasındaki sayıları ekleyiniz.

3. Adım: ComboBox nesnesi içinden sınıf bilgisi seçildiğinde, seçilen sınıfın hangi eğitim-öğretim kademesine ait olduğunu mesaj veren programı Görsel 2.9'da görüldüğü gibi tasarlayıp yazınız.



Görsel 2.9: Ses ayarı

```
private void comboBox1_SelectedValueChanged(object sender, EventArgs e)
{
    byte sinif;
    sinif = Convert.ToByte(comboBox1.Text);
    if (sinif > 0 && sinif < 5)
    {
    MessageBox.Show("İlkokul kademesi");
    }
    else if (sinif > 4 && sinif < 9)
    {
    MessageBox.Show("Ortaokul kademesi");
    }
    else if (sinif > 8 && sinif < 13)
    {
    MessageBox.Show("Lise kademesi");
     }
}
```

Not ComboBox nesnesinin değeri değiştiğinde kodların çalışması için SelectedValueChanged olay metodu oluşturulur.

Sıra Sizde

TextBox nesnelerine kilo ve boy bilgileri girilen kişinin vücut kütle indeksini hesaplayıp, çıkan sonuca göre vücut kütle indeksinin hangi kategoride olduğunu mesaj veren programı Görsel 2.10'da görüldüğü gibi tasarlayıp yazınız.

Görsel 2.10: Vücut kütle indeksi uygulaması

Tablo 2.3: Vücut Kütle İndeksi

Operatör	Anlamı
<	Küçükse
>	Büyükse
==	Eşitse
<=	Küçük veya Eşitse
>=	Büyük veya Eşitse
!=	Eşit Değilse

Not Vücut kütle indeksi; bir kişinin kilogram cinsinden ağırlığının, metre cinsinden boy uzunluğunun karesine bölünmesiyle hesaplanır. Örneğin ağırlığı 78 kg ve boyu 1.78 m olan biri için vücut kütle indeksi=kg/m^2 formülünden 23,4 kg/m^2 olarak hesaplanacaktır. Tablo 2.3'e göre 23,4 kg/m^2 indeksine sahip kişi ideal kilosundadır.

Sıra Sizde

TextBox nesnelerine girilen üç sayıdan hangisinin en büyük sayı, hangisinin en küçük sayı olduğunu bulan programı yazınız.

2.2.2. OR(||) Operatörü

Şart ifadelerinin en az bir tanesinin sağlanması gerektiğinde "or" operatörü kullanılır. Şart ifadelerinin hiçbiri sağlanmadığında if yapısına bağlı kod bloku çalışmaz.

```
if(yas > 65 || engel_durumu == true)
{
// kodlarınız
}
```

Not : Yeni bir | | operatörü kullanılarak şart ifadelerinin sayısı artırılabilir.

14. Uygulama

1. Adım: Form üzerine 1 adet TextBox, 1 adet Label, 1 adet Button, 1 adet GroupBox ve 5 adet RadioButton nesnesini ekleyiniz.

2. Adım: Form nesnesinin rengini "ActiveCaption" yapınız.

3. Adım: TextBox nesnesine toplam tutarı girilen bir alışverişin RadioButton nesneleri ile işaretlenen ödeme şekli 2 taksit veya 3 taksit ise ödenecek toplam tutara %5 ek fiyat, ödeme şekli 4 taksit veya 5 taksit ise ödenecek toplam tutara %10 ek fiyat ilave eden programı Görsel 2.11'de görüldüğü gibi tasarlayıp yazınız.

```
private void button1_Click(object sender, EventArgs e)
    {
    double tutar;
    tutar = Convert.ToDouble(textBox1.Text);
    if(radioButton2.Checked==true || radioButton3.Checked == true)
    {
    tutar = tutar + (tutar * 0.05);
    }
    if (radioButton4.Checked == true || radioButton5.Checked == true)
   {
   tutar = tutar + (tutar * 0.10);
    }
   MessageBox.Show("Ödenecek Toplam Tutar:" + tutar.ToString() + "TL");
}
```

Görsel 2.11: Kasa uygulaması

15. Uygulama

http://kitap.eba.gov.tr/KodSor.php?KOD=21075

1. Adım: Form üzerine 1 adet ListBox ve 1 adet Button nesnesini ekleyiniz.

2. Adım: ListBox nesnesinin Items özelliğine "Mouse, Yazıcı, Klavye, Hoparlör, Kamera, Tarayıcı, Projeksiyon" değerlerini ekleyiniz.

3. Adım: ListBox nesnesi içinden seçilen bir bilgisayar parçasının giriş donanım birimi mi, çıkış donanım birimi mi olduğunu mesaj veren programı Görsel 2.12'de görüldüğü gibi tasarlayıp yazınız.

Görsel 2.12: Bilgisayar donanım birimleri uygulaması

```
private void button1_Click(object sender, EventArgs e)
 {
    string secim;
    secim = listBox1.SelectedItem.ToString();
    if(secim=="Mouse" || secim=="Klavye" || secim == "Kamera" || secim=="Tarayıcı")
    {
     MessageBox.Show("Bu parça, giriş birimidir.");
     }
    if (secim == "Yazıcı" || secim == "Hoparlör" || secim == "Projeksiyon")
    {
    MessageBox.Show("Bu parça, çıkış birimidir.");
     }
 }
```

2.2.3. Mantıksal Operatör Önceliği

Karar ifadelerinde ve(&&) operatörü veya(||) operatörüne göre daha öncelikli işleme alınır.

16. Uygulama

1. Adım: Form üzerine 2 adet GroupBox, 2 adet RadioButton, 1 adet ListBox ve 1 adet Button nesnesi ekleyiniz.

2. Adım: ListBox nesnesinin Items özelliğine "MP4, JPG, MOV, PNG" değerlerini ekleyiniz.

3. Adım: ListBox nesnesi içinden seçilen dosya uzantısı ile RadioButton nesnelerinden seçilen dosya türü bilgisinin eşleşmesini kontrol eden programı Görsel 2.13'te görüldüğü gibi tasarlayıp yazınız.

Görsel 2.13: Dosya uzantısı ve türünü değiştirme ayarları

Not : On altıncı uygulamada ve(&&) operatörü öncelikli olarak işleme alınacağı için veya(| |) operatörleri ile oluşturulan şart ifadeleri ayrı bir parantez içine alınmıştır. Aksi durumda program hatalı çalışacaktır.

```
private void button1_Click(object sender, EventArgs e)
{
    string secim;
    bool cevap1, cevap2;
    secim = listBox1.SelectedItem.ToString();
    cevap1 = radioButton1.Checked;
    cevap2 = radioButton2.Checked;
     if(cevap1==true && (secim=="MP4" || secim=="MOV" ))
     {
     MessageBox.Show("Cevabınız Doğru");
      }
     if (cevap1 == false && (secim == "MP4" || secim == "MOV"))
     {
     MessageBox.Show("Cevabınız Yanlış");
     }
     if (cevap2 == true && (secim == "JPG" || secim == "PNG"))
     {
     MessageBox.Show("Cevabınız Doğru");
     }
    if (cevap2 == false && (secim == "JPG" || secim == "PNG"))
    {
     MessageBox.Show("Cevabınız Yanlış");
      }
}
```

Sıra Sizde

On altıncı uygulamayı veya(| |) operatörlerini ayrı bir parantez içine almadan çalıştırınız. Oluşacak hatalı mesajları arkadaşlarınızla paylaşınız.

2.2.4. NOT(!) Operatörü

NOT(!) operatörü, şart ifadesinin alacağı sonuç true ise false; şart ifadesinin alacağı sonuç false ise true şeklinde değiştirir. Şart ifadesinin önüne değil(!) operatörü konularak kullanılır.

```
if (!radioButton1.Checked == true)
   {
   // kodlarınız
   }
```

Not : Örnek kodda değil(!) operatörü kullanıldığı için radioButton1 nesnesi işaretli değilken kod çalışacaktır.

```
if (!(yas>=15 && yas<=65))
 {
 // kodlarınız
 }
```

Not : Örnek kodda yas değişkeni şart ifadesinde belirtilen aralığın dışında ise kodlar çalışacaktır (Örneğin yas=32 olduğunda kodlar çalışmayacak, yas=70 olduğunda kodlar çalışacaktır.).

17. Uygulama

1. Adım: Form üzerine 2 adet RadioButton nesnesi ve 1 adet Button nesnesi ekleyiniz.

2. Adım: Form nesnesinin arka plan rengini gri yapınız.

3. Adım: Button nesnesine tıklandığında üstteki RadioButton işaretliyse Form nesnesinin arka plan rengini siyah, alttaki RadioButton işaretliyse arka plan rengini beyaz yapan programı Görsel 2.14'te görüldüğü gibi tasarlayıp yazınız.

Görsel 2.14: Arka plan ayarları

```
private void button1_Click(object sender, EventArgs e)
{
      if (radioButton1.Checked == true)
      {
      this.BackColor = Color.Black;
      radioButton1.ForeColor = Color.White;
      radioButton2.ForeColor = Color.White;
      }
      if (!radioButton1.Checked == true)
      {
      this.BackColor = Color.White;
      radioButton1.ForeColor = Color.Black;
      radioButton2.ForeColor = Color.Black;
      }
}
```

Not : On yedinci uygulamada if (!radioButton1.Checked == true) kodu yerine if (radioButton1.Checked == false) kodu veya else yapısı da kullanılabilir.

Sıra Sizde

Bir arkadaşınızla birlikte aşağıdaki şart ifadelerini yeni bir proje dosyası açarak deneyiniz ve çıkan sonuçlara göre kodların hangi durumlarda çalışacağını açıklama kısmına yazınız.

Şart İfadesi	Açıklama
if (!(Stok>=10))	Stok değişkeninin değeri 10'dan küçük olduğunda çalışır.
if (!(sayi > 0 && sayi <10))	
if (!(cinsiyet=="erkek"))	
if (!(numara%2==1))	
if (!(renk=="sarı") && numara%10==2)	
if (textBox1.Text != "Murat")	

2.3. DÖNGÜLER

Programda belirli kodların tekrar tekrar çalıştırılmasını sağlayan yapılara döngü denir. Duruma göre aynı kod 2 kez çalıştırılabileceği gibi 200 kez hatta 2000 kez de çalıştırılabilir. Böyle durumlarda program yazmak zorlaşacak, zaman alacak ve programın kod yapısı karmaşıklaşacaktır. Örneğin 1000 adet ürün arasından girilen barkod numarasına ait ürünün bilgilerini getirmek için 1000 adet if komutu kullanmak yerine döngü ifadesi içinde sadece bir tane if komutu kullanılabilir.

2.3.1. Sayaçlar

Bir değişkene bağlı değeri farklı aralıklarla artırmak, azaltmak, katlamak veya bölmek gerekebilir. Böyle durumlarda sayaç adı verilen değişkene değer atama yöntemleri kullanılır (Görsel 2.15). Tablo 2.4'te sayaçların kullanımına örnekler verilmiştir.

Tablo 2.4: Sayaçlar

Basit Atamayla Sayaç İfadesi	Bileşik Atamayla Açıklama
x=x+1;	x+=1;
x=x-2;	x-=2;
x=x*3;	x*=3
x=x/5;	x/=5;

Görsel 2.15: Döngüler

18. Uygulama

1. Adım: Form üzerine 2 adet Button ve 1 adet TextBox nesnesi ekleyiniz.

2. Adım: Eksi bir yazılı Button nesnesine tıklandığında TextBox nesnesindeki sayıyı 1 azaltan, artı bir yazılı Button nesnesine tıklandığında TextBox nesnesindeki sayıyı 1 artıran programı Görsel 2.16'daki gibi tasarlayıp yazınız.

Görsel 2.16: Sayaçlar

```
private void button1_Click(object sender, EventArgs e)
{
int sayi;
sayi = Convert.ToInt32(textBox1.Text);
sayi = sayi - 1;
textBox1.Text = sayi.ToString();
}
private void button2_Click(object sender, EventArgs e)
{
int sayi;
sayi = Convert.ToInt32(textBox1.Text);
sayi = sayi + 1;
textBox1.Text = sayi.ToString();
}
```

2.3.2. Artırma ve Azaltma Operatörleri

Artırma ve azaltma operatörleri döngü yapılarında çok sık kullanılır. Döngü değişkeninin değeri birer birer artıyor veya birer birer azalıyorsa sayaç yerine pratik bir kullanıma sahip olan artırma ve azaltma operatörleri tercih edilebilir (Tablo 2.5).

Tablo 2.5: Sayaçlar

Operatör	Açıklama
x++;	Değişkenin değerini 1 artırır.
x--;	Değişkenin değerini 1 azaltır.
x=x*3;	x*=3
x=x/5;	x/=5;

19. Uygulama

1. Adım: Form üzerine 2 adet PictureBox ve 4 adet Button nesnesi ekleyiniz.

2. Adım: PictureBox nesnelerine at resmi yerleştiriniz.

3. Adım: Normal yazılı Button nesnelerine tıklandığında PictureBox nesnesinin Left özelliğini 10 artırarak, Hızlı yazılı Button nesnesine tıklandığında PictureBox nesnesinin Left özelliğini 25 artırarak PictureBox nesnelerini hareket ettiren programı, Görsel 2.17'deki gibi tasarlayıp yazınız (Herhangi bir nesne, Left özelliğinin değeri değiştirilerek Form nesnesi üzerinde sola veya sağa kaydırılabilir.).

Görsel 2.17: Artırma operatörü

```
private void button1_Click(object sender, EventArgs e)
 {
 pictureBox1.Left = pictureBox1.Left + 10;
 }
 private void button2_Click(object sender, EventArgs e)
 {
 pictureBox1.Left = pictureBox1.Left + 25;
     }
 private void button3_Click(object sender, EventArgs e)
 {
 pictureBox2.Left = pictureBox2.Left + 10;
 }
 private void button4_Click(object sender, EventArgs e)
 {
 pictureBox2.Left = pictureBox2.Left + 25;
 }
```

2.3.3. For Döngüsü

Genellikle kodların tekrar sayısı belli olduğunda for döngüsü kullanılır. Döngü için tanımlanan şart ifadesi her sağlandığında döngüdeki kodlar tekrar çalışır. For döngüsünün kaç kez çalışacağını belirlemek oldukça basittir.

```
for (Döngü değişkeni başlangıç değeri; Döngü şart ifadesi; Döngü değişkeni sayacı)
 {
 //kodlarınız
 }
```

Yandaki örnek kodda "i" değişkeninin alacağı her değer için aynı kodlar çalışır. "i" değişkenindeki değer artışı "i++" sayacı (artış operatörü) ile sağlanır. "i" değişkeni sırasıyla 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 değerlerini alır ve böylece döngü içindeki kodlar toplam 10 kez çalışır. "i" değişkeninin değeri 10 olduğunda "i<10" şart ifadesi sağlanamadığı için döngü durur (Döngü değişkeni ikişer ikişer artmış olsaydı döngü içindeki kodlar 5 kez çalışacaktı.).

```
for (int i = 0; i < 10; i++)
{
//kodlarınız
}
```

20. Uygulama

1. Adım: Form üzerine 1 adet ListBox nesnesi ekleyiniz.

2. Adım: ListBox nesnesinin içine 7 tane "Bilişim Teknolojileri" metni ekleyen programı yazınız.

```
private void button1_Click(object sender, EventArgs e)
  {
      for (int i = 0; i < 7; i++)
      {
      listBox1.Items.Add("Bilişim Teknolojileri");
      }
  }
```

Not For döngüsü ile pratik bir şekilde ListBox nesnesine elemanlar eklenir.

```
private void button1_Click(object sender, EventArgs e)
{
listBox1.Items.Add("Bilişim Teknolojileri");
listBox1.Items.Add("Bilişim Teknolojileri");
listBox1.Items.Add("Bilişim Teknolojileri");
listBox1.Items.Add("Bilişim Teknolojileri");
listBox1.Items.Add("Bilişim Teknolojileri");
listBox1.Items.Add("Bilişim Teknolojileri");
listBox1.Items.Add("Bilişim Teknolojileri");
}
```

Not Döngü kullanılmadan uygulama yapıldığında ListBox'a eklenmek istenen elemanların hepsini tek tek yazmak gereklidir.

21. Uygulama

1. Adım: Form üzerine 1 adet ListBox nesnesi ekleyiniz.

2. Adım: ListBox nesnesinin içine 1'den 10'a kadar olan sayıları ekleyen programı yazınız.

```
private void button1_Click(object sender, EventArgs e)
 {
    for (int i = 1; i <= 10; i++)
    {
    listBox1.Items.Add(i);
    }
 }
```

Not Döngü değişkeninin başlangıç değeri "1", bitiş değeri (şart ifadesindeki değeri) "10", artış miktarı da "1"dir. Böylece "i" değişkeni sırasıyla 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 değerlerini alır. Bunlar, ListBox nesnesine eklenecek değerlerdir.

22. Uygulama

1. Adım: Form üzerine 1 adet ComboBox nesnesi ekleyiniz.

2. Adım: ComboBox nesnesinin içine 6'dan 16'ya kadar olan sayıları ekleyen programı yazınız.

```
private void button1_Click(object sender, EventArgs e)
 {
     for (int i = 6; i <=16; i++)
     {
     comboBox1.Items.Add(i);
     }
 }
```

Not Döngü değişkeninin başlangıç değeri "6", bitiş değeri (şart ifadesindeki değeri) "16", artış miktarı da "1"dir. Böylece "i" değişkeni sırasıyla 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 değerlerini alır.

23. Uygulama

1. Adım: Form üzerine 1 adet ComboBox nesnesi ekleyiniz.

2. Adım: ComboBox nesnesinin içine 100 ile 150 arasında 5'in katı olan sayıları ekleyen programı yazı-

```
private void button1_Click(object sender, EventArgs e)
 {
     for (int i = 100; i <=150; i=i+5)
     {
     comboBox1.Items.Add(i);
     }
 }
```

Not Döngü değişkeninin başlangıç değeri "100", bitiş değeri (şart ifadesindeki değeri) "150", artış miktarı da "5"tir. Böylece "i" değişkeni sırasıyla 100, 105, 110, 115, 120, 125, 130, 135, 140, 145, 150 değerlerini alır.

Sıra Sizde

ListBox nesnesinin içine sırasıyla 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110 sayılarını ekleyen programı yazınız.

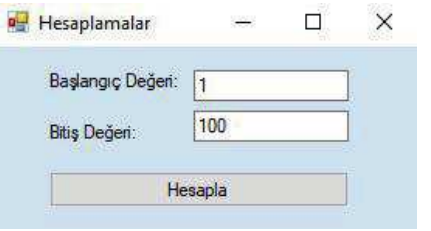
Sıra Sizde

ListBox nesnesinin içine sırasıyla 110, 100, 90, 80, 70, 60, 50, 40, 30, 20, 10 sayılarını ekleyen programı yazınız.

24. Uygulama

1. Adım: Form üzerine 2 adet Label, 2 adet TextBox ve 1 adet Button nesnesi ekleyiniz.

2. Adım: Button nesnesine tıklandığında TextBox nesneleri içine girilen başlangıç ve bitiş değerleri arasındaki sayıları toplayarak sonucu mesaj veren programı, Görsel 2.18'de görüldüğü gibi tasarlayıp yazınız.



Görsel 2.18: Hesaplamalar

```
private void button1_Click(object sender, EventArgs e)
{
int sayi1, sayi2, toplam;
sayi1 = Convert.ToInt32(textBox1.Text);
sayi2 = Convert.ToInt32(textBox2.Text);
toplam = 0;
for (int i = sayi1; i <=sayi2 ; i++)
{
toplam = toplam + i;
}
MessageBox.Show("Sayıların Toplamı=" + toplam.ToString());
}
```

Not : Yirmi dördüncü uygulamada döngünün başlangıç ve bitiş değerleri için değişkenler kullanılmıştır.

25. Uygulama

1. Adım: Form üzerine 1 adet ListBox ve 1 adet Button nesnesi ekleyiniz.

2. Adım: 10 sayısının 0'dan 4'e kadar olan kuvvetlerini, ListBox nesnesi içine ekleyen programı Görsel 2.19'da görüldüğü gibi tasarlayıp yazınız.

Görsel 2.19: Kuvvet alma

```
private void button1_Click(object sender, EventArgs e)
{
     double kuvvet;
     for (int i = 0; i < 5; i++)
    {
    kuvvet = Math.Pow(10, i);
    listBox1.Items.Add(kuvvet);
    }
}
```

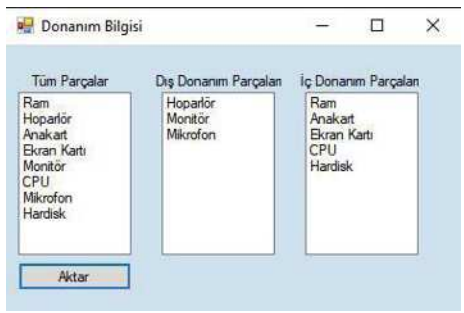
Not : Math.Pow fonksiyonu bir sayının istenilen kuvvetini almak için kullanılır. Bu fonksiyon, double veri türünde geriye değer döndürür.

26. Uygulama

1. Adım: Form üzerine 3 adet ListBox, 3 adet Label ve 1 adet Button nesnesi ekleyiniz.

2. Adım: İlk ListBox nesnesinin Items özelliğine "Ram, Hoparlör, Anakart, Ekran Kartı, Monitör, CPU, Mikrofon, Harddisk" değerlerini ekleyiniz.

3. Adım: Button nesnesine tıklandığında listBox1 nesnesi içinde karışık bir şekilde listelenmiş bilgisayar parçalarını dış donanım ve iç donanım birimleri şeklinde ayırıp listBox2 ve listBox3 nesnelerine aktaran programı, Görsel 2.20'de görüldüğü gibi tasarlayıp yazınız.



Görsel 2.20: Donanım bilgisi

```
private void button1_Click(object sender, EventArgs e)
 {
     for (int i = 0; i < listBox1.Items.Count; i++)
     {
         if(listBox1.Items[i].ToString() == "Hoparlör" ||
          listBox1.Items[i].ToString() == "Mikrofon" ||
          listBox1.Items[i].ToString() == "Monitör")
          {
          listBox2.Items.Add(listBox1.Items[i]);
          }
          else
          {
          listBox3.Items.Add(listBox1.Items[i]);
          }
      }
 }
```

Not : listBox1.Items.Count kodu, listBox nesnesinin içindeki item sayısını verir. Yirmi altıncı uygulamada listBox1 nesnesinde 8 tane parça ismi ekli olduğu için listBox1.Items.Count kodu ile 8 sayısı elde edilir.

Sıra Sizde

Rastgele 20 adet sayıyı listBox1 nesnesi içine ekleyiniz. Butona tıkladığınızda listBox1 nesnesindeki tek sayıları listBox2 nesnesine, çift sayıları ise listBox3 nesnesine aktaran programı yazınız.

2.3.4. While Döngüsü

Döngü için tanımlanan şart ifadesi sağlanıyorsa döngü çalışmaya başlar. Şart ifadesi sağlandığı sürece while döngüsü çalışmaya devam eder.

```
while (Şart ifadesi);
{
//kodlar
}
```

27. Uygulama

1. Adım: Form üzerine ListBox nesnesi ekleyiniz.

2. Adım: ListBox nesnesi içine 1'den 10'a kadar olan sayıları ekleyen programı yazınız.

```
int say = 1;
while (say<=10)
{
listBox1.Items.Add(say);
say++;
}
```

Not : **While** ve **do-while** döngülerinde döngü değişkeni, for döngüsünden farklı olarak döngünün dışında tanımlanır ve değişkenin değeri döngünün içinde değiştirilir.

Sıra Sizde

ListBox nesnesi içine 20 tane farklı sayı ekleyiniz. Button nesnesine tıklandığında ListBox nesnesindeki ilk sayıdan başlayarak sayıları toplayan programı, while döngüsü kullanarak yazınız.

28. Uygulama

0'dan başlayarak sırasıyla sayıları toplayan ve sayıların toplamı 1000'den fazla olduğunda döngünün kaç kez çalıştığını bulan programı yazınız.

```
int dongu_say = 1;
int toplam = 0;
while (toplam<=1000)
{
toplam = toplam + dongu_say;
dongu_say += 1;
}
MessageBox.Show("Döngü toplam " + dongu_say.ToString() + " kez çalıştı");
```

2.3.5. Do-while Döngüsü

Döngü için tanımlanan şart ifadesi sağlanmasa da do-while döngüsü en az bir kez çalışır çünkü while döngüsünde şart ifadesi döngünün başlangıcındayken do-while döngüsünde şart ifadesi döngünün sonundadır. Do-while döngüsü de diğer döngüler gibi şart sağlandığı sürece çalışmaya devam eder.

```
do{//kodlar
   }while (Şart ifadesi);
```

29. Uygulama

1. Adım: Form üzerine ListBox nesnesi ekleyiniz.

2. Adım: ListBox nesnesi içine 1'den 10'a kadar olan sayıları ekleyen programı yazınız.

```
int say = 1;
 do
 {
 listBox1.Items.Add(say);
 say++;
 }while(say<=10);
```

Do-while döngüsü şart sağlanmasa bile içindeki kodları bir kez çalıştırır.

do-while	while

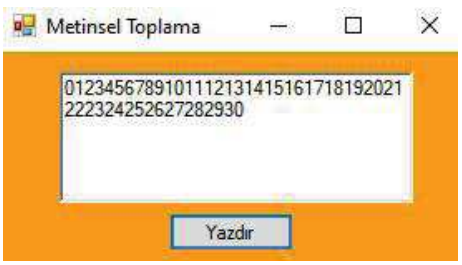
int say = 1
do
{
listBox1.Items.Add(say);
say++;
} while (say > 5);
	int say = 1;
	while (say > 5)
	{
	listBox1.Items.Add(say);
	say++;
	}

Not : While döngüsünde şart ifadesi sağlanmadığı için listBox1 nesnesine hiçbir sayı eklenmeyecektir. Do-while döngüsünde şart sağlanmasa bile listBox1 nesnesine bir tane eleman eklenecektir.

30. Uygulama

1. Adım: Form üzerine 1 adet RichTextBox, 1 adet Button nesnesi ekleyiniz.

2. Adım: Button nesnesine tıklandığında 0'dan 30'a kadar olan sayıları yan yana RichTextBox nesnesine aktaran programı do-while döngüsü kullanarak, Görsel 2.21'deki gibi tasarlayıp yazınız.



Görsel 2.21: Metinsel toplama

```
private void button1_Click(object sender, EventArgs e)
{
   int say=0;
   do{

   richTextBox1.Text = richTextBox1.Text + say.ToString();
   say++;
   } while (say<=30);
}
```

2.3.6. Döngüyü Kesme (Durdurma)

Döngüyü kesme komutu "break;" döngü tamamlanmadan döngüden çıkmak için kullanılır. "break;" komutundan sonra döngüye ait hiçbir kod çalışmaz ve program, döngü ifadesinden sonraki kod satırı ile çalışmaya devam eder.

Not : Döngü türleri için ayrı ayrı verilen örnek kodlara göre "a" döngü değişkeni en son 5 değerini alır, "break;" kesme komutu kullanıldığı için döngü tamamlanmadan durur ve döngü 10 kez çalışması gerekirken 6 kez çalışır.

For Döngüsü

```
for (int a = 0; a < 10; a++)
{
    MessageBox.Show("Döngü Çalışıyor.");
    if (a == 5)
    {
    MessageBox.Show("Döngü Durdu.");
    break;
    }
}
```

While Döngüsü

```
int a = 0;
while (a<10)
 {
  MessageBox.Show("Döngü Çalışıyor.");
  if (a == 5)
    {
     MessageBox.Show("Döngü Durdu.");
     break;
    }
  a++;
  }
```

Do-While Döngüsü

```
int a = 0;
do
{
     MessageBox.Show("Döngü Çalışıyor.");
     if (a == 5)
     {
     MessageBox.Show("Döngü Durdu.");
      break;
      }
      a++;
}while (a<10);
```

31. Uygulama

1. Adım: Form nesnesi üzerine 2 adet Label, 1 adet ListBox, 1 adet TextBox ve 1 adet Button nesnesi ekleyiniz.

2. Adım: ListBox nesnesinin Items özelliğine 10 adetten fazla isim giriniz.

3. Adım: TextBox nesnesi içine girilen müşteri adını ListBox nesnesi içinde arayarak, eşleşen ad varsa "Aradığınız müşteri bulundu!" şeklinde mesaj veren programı Görsel 2.22'de görüldüğü gibi tasarlayıp yazınız.

Görsel 2.22: Müşteri arama

```
string ad = textBox1.Text;

 for (int i = 0; i < listBox1.Items.Count; i++)
 {
 if (listBox1.Items[i].ToString() == ad)
   {
 MessageBox.Show(listBox1.Items[i].ToString()  + " adlı müşteri bulundu!");
 break;
    }
 }
```

Not : Uygulamada "....adlı müşteri bulundu!" mesajı verildikten sonra "break;" komutu çalıştığı için döngü kesintiye uğrar ve sonlanır. Döngü içinde "break;" komutu kullanılmasaydı müşterinin adı bulunduktan sonra döngüde yapılan tüm işlemler gereksiz olacaktı. Bu gereksiz işlemler, programın çalışma performansını etkiler. 3000 tane müşteri adının kayıtlı olduğu bir listede aranan müşteri listenin 15. sırasında bulunmuş ise geri kalan 2985 tane kaydı kontrol etmek hem gereksizdir hem de programın performans kaybına sebep olur.

Sıra Sizde

1. Otuz birinci uygulamada for döngüsü ile düzenlenmiş döngü kodlarını **while** ve **do-while** döngülerini kullanarak ayrı ayrı yazınız.

2. 1 ile 100 arasında 3'ün katı olan sayıları toplarken sayıların toplamı 200'ü geçince sadece bir kez "Li¬mit aşıldı." mesajı veren programı yazınız.

```
int .....= 0;
 for(int say = 0; say < 100; say = say +.....)
 {
       topla = topla + say;
       if (.....)
       {
       MessageBox.....
        .....
        }
 }
```

2.3.7. Döngüyü Devam Ettirme

Döngü içinde "continue;" komutu çalıştıktan sonra diğer kodlar döngünün o andaki adımı için çalışmaz ve döngü bir sonraki adıma geçer. Böylece döngü isteğe bağlı bir şekilde belli adımları atlayarak çalışır.

Aşağıdaki döngü ifadelerinde "a" değişkeninin değeri 5'e veya 10'a eşit olduğunda döngü bir sonraki değerini alır.

```
for (int a = 1; a < 15; a++)
{
  if (a==5 || a==10 )
  {
  continue;
   }
  //kodlar
}
```

```
int a = 1;
while (a<15)
{
   //kodlar
   if (a==5 || a==10)
   {
   continue;
    }
   a++;
}
```

```
int a = 1;
do{
   //kodlar
   if (a==5 || a==10)
   {
   continue;
    }
   a++;
}while (a<15);
```

Sıra Sizde

Aşağıdaki kodlara göre listBox1 nesnesi içine hangi sayılar eklenir?

```
for (int i = 1; i <= 10; i++)
{
  if (i < 7)
  {
  continue;
   }
   listBox1.Items.Add(i);
}
```

Cevap:

```
for (int i = 1; i <= 10; i=i+2)
{
  listBox1.Items.Add(i);
   if (i >= 7)
  {
  continue;
   }
}
```

Cevap:

2.4. HATA AYIKLAMA

Kod editörü derleyicisi, programın kodları yazım kurallarına uygun olmadığında Hata Listesi panelinde hatalı kodları gösterir (Görsel 2.23). Bazı hatalar, program çalıştıktan sonra oluşur ve programı durdurur. Örneğin programda TextBox nesnesine girilen sayı ile işlem yapılıyorsa fakat TextBox nesnesine sayı yerine harf girilmişse program hata verip duracaktır. Bu hata, kullanıcının yanlış veri girişinden kaynaklandığı için derleyici, bu tip hataları yakalayıp Error List panelinde gösteremez. Bu tip hataların programı durdurmaması için önlem olarak **try-catch-finally** hata ayıklama blokları kullanılmalıdır.

Görsel 2.23: Hata ayıklama

Kodlar çalıştırıldığında textBox1 nesnesine sayı yerine harf girilmişse program Görsel 2.24'teki gibi hata verecek ve durdurulacaktır. Bu tip hatalar, özel durumlarda oluşan hatalardır ve program çalıştırılmadan önce derleyici tarafından tespit edilemez.

```
private void button1_Click(object sender, EventArgs e)
{
    int sayi1,karesi;
    sayi1 = Convert.ToInt32(textBox1.Text);
    karesi = sayi1 * sayi1;
    MessageBox.Show(karesi.ToString());
```

Özel Durum İşlenmedi

System.FormatException: 'Giriş dizesi doğru biçimde değildi.'

Bu özel durum özgün olarak bu çağrı yığınında oluşturuldu:
[Dış Kod]

Görsel 2.24: Hatalı kod-1

2.4.1. Try-Catch-Finally Bloku

Program çalıştırıldığında hata meydana gelme olasılığı olan kodlar **try** bloku içinde yazılır. Try bloku içine yazılan kodlarda hata meydana gelirse program **try** blokundan çıkarak **catch** bloku içindeki kodları çalıştırır. Böylece program hata vermez ve çalışmaya devam eder.

Try bloku içine yazılan kodlarda bir hata meydana gelmezse **catch** blokundaki kodlar çalışmaz fakat **try** blokunda hata meydana gelse de gelmese de **finally** blokundaki kodlar çalışır. **Finally** blokunu kullanmak zorunlu değildir, tercihe bağlıdır.

32. Uygulama

1. Adım: Form üzerine ButtonTextBox nesnesi ekleyiniz.

2. Adım: Button nesnesinin Click olay metodunu oluşturunuz.

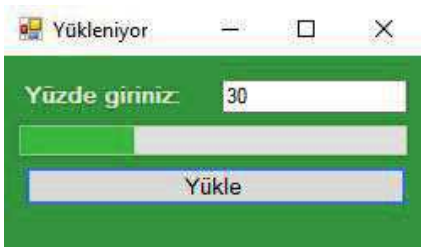
3. Adım: Aşağıdaki kodları Click olay metodunun içine yazınız.

```
int sayi1,sayinin_karesi;
  try
  {
  sayi1 = Convert.ToInt16(textBox1.Text);
  sayinin_karesi = sayi1 * sayi1;
  MessageBox.Show(sayinin_karesi.ToString());
  }
  catch
  {
  MessageBox.Show("Hatalı giriş yaptınız!");
  }
  finally
  {
  // Hata olsa da olmasa da çalışacak kodlar
  }
```

Not Kullanıcı, otuz ikinci uygulamadaki kodlar çalıştırıldıktan sonra textBox1 nesnesine sayı dışında bir değer girerse program hata verir veya sayinin_karesi değişkeninin alacağı değer int veri tipinin kapasite sınırlarının dışında olursa program yine hata verir. Her iki hata durumunda da catch blokundaki kodlar çalışacaktır.

Sıra Sizde

TextBox nesnesi içine girilen yüzde değerine göre ProgressBar nesnesinin değerini değiştiren programı hata ayıklama kodlarını kullanarak, Görsel 2.25'te görüldüğü gibi tasarlayıp yazınız.



Görsel 2.25: Yüzde uygulaması

```
private void button1_Click(object sender,
{
    byte yuzde;
    yuzde = Convert.ToByte(textBox1.Text);
    progressBar1.Value = yuzde;
}
```

Görsel 2.26: Hatalı kod

Not ProgressBar nesnesi 0 ile 100 arasındaki değerleri alır. Program, hata ayıklama kodları olmadan yazılırsa Görsel 2.26'daki gibi hata meydana gelecek ve program duracaktır.

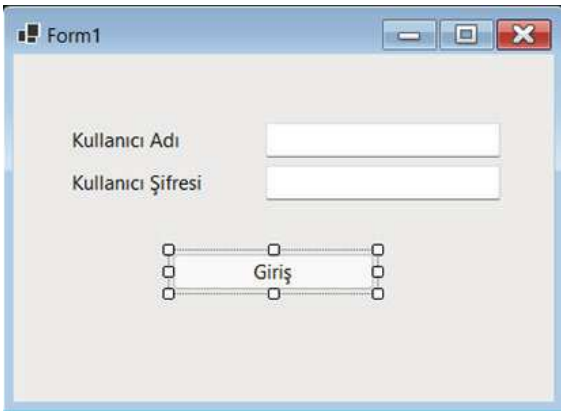
33. Uygulama

1. Adım : Form üzerine 2 adet Label 2 adet TextBox ögesi ekleyiniz. Label'ların text özelliğine **Görsel'deki** ifadeleri giriniz.

2. Adım : TextBox ögelerinin name özelliğine sırasıyla "textBox_KullaniciAdi" ve "textBox_KullaniciSifresi" isimlerini giriniz.

3. Adım : Form üzerine Button ögesi ekleyiniz. Button'un name özelliğine "button_Giris" , text özelliğine "Giriş" değerlerini yazınız

4. Adım : Aşağıdaki kodları Buttonun Click olay metotunun içerisine yazalım.



Görsel 2.27: Kullanıcı Giriş

```
string kullaniciAdi;
        long parola;

        try
        {
            kullaniciAdi = textBox_KullaniciAdi.Text.ToString();
            parola=long.Parse(textBox_KullaniciSifresi.Text.ToString());
            MessageBox.Show("Giriş Başarılı. Hoşgeldiniz " + kullaniciAdi);
        }
        catch (Exception)
        {
            MessageBox.Show("Şifreniz Sadece Sayılardan Oluşmalıdır. Tekrar Deneyiniz.");
            textBox_KullaniciSifresi.Text = "";
        }
        finally
        {

        }
```

Burada kullanıcı adı girilen textboxta herhangi bir kısıtlama bulunmamaktadır. String tanımlanmış bu özellikte her ifadenin girilmesi açık bırakılmıştır. Try-catch bloklarında, kullanıcı adı girişi sebebiyle herhangi bir sorun karşılaşma oranı sıfırdır. Fakat parola textboxı long olarak tanımlanmıştır. Bu nedenle herhangi bir metinsel ifade veya long veri tipinden daha uzun olan sayısal ifadeleri kabul etmeyecektir. Try-catch bloğu içerisinde yazılmazsa program hata verecektir. Bu nedenle yazılan try – catch bloğunda ise eğer parola long ifadeden daha uzun sayısal veri olursa veya sayısal ifadeden başka ifadeler yazılırsa Catch kısmı çalışacak ve kullanıcıya MessageBox ile uyarı verecektir. Sonrasında ise yazılan kod ile parola textboxı sıfırlanacak içindeki veri silinecektir. Eğer uygun bir parola yazılırsa (long veri tipi sınırlarında sayısal ifade) "Giriş Başarılı. Hoşgeldiniz" ve yanında da kullanıcı adını yazan bir MessageBox ifadesi belirecektir.

ÖLÇME VE DEĞERLENDİRME

A) Aşağıdaki cümlelerde parantez içine yargılar doğru ise "D", yanlış ise "Y" yazınız.

1. ( ) <,>,==,<=,>=,!= karakterleri mantıksal operatörlerdir.

2. ( ) Üçten fazla if ifadesi iç içe kullanılmaz.

3. ( ) Şart ifadesi sağlanmadığında çalışacak kodlar "else" blokunun içine yazılabilir.

4. ( ) Karar ifadelerinde ve(&&) operatörü veya(||) operatöründen öncelikli işleme alınır.

5. ( ) Döngü şart ifadesi sağlanmasa da for döngüsü en az bir kez çalışır.

B) Aşağıda boş bırakılan kutucuklara soru kökünde istenen kodları uygun şekilde yazınız.

6. Eğer "a" değişkeni "b" ve "c" değişkeninden büyükse ifadesini kod olarak yazınız.

7. Eğer "a" değişkeni "b" veya "c" değişkeninden küçükse ifadesini kod olarak yazınız.

8. Aşağıdaki kodlar çalıştırıldığında comboBox1 nesnesine eklenecek değerler nelerdir?

```
for (int i = 1; i <= 20; i = i + 1)
{
        if (i % 3 == 0)
            {comboBox1.Items.Add(i); }
        if (i == 17)
            {comboBox1.Items.Add(i); }
}
```

9. Aşağıdaki kodlar çalıştırıldığında comboBox1 nesnesine eklenecek değerler nelerdir?

```
for (int i = 1; i <= 15; i = i + 1)
{
      if (i % 3 == 0)
          { continue; }
          comboBox1.Items.Add(i);
}
```

10. Aşağıdaki kodlar çalıştırıldığında oluşacak mesajları sırasıyla yazınız.

```
int sayac = 0,sonuc=1;
while (sayac<6)
{
sayac = sayac+1;
sonuc = sonuc * sayac;
MessageBox.Show(sayac.ToString()+ ".tur sonuç=" + sonuc.ToString());
}
```

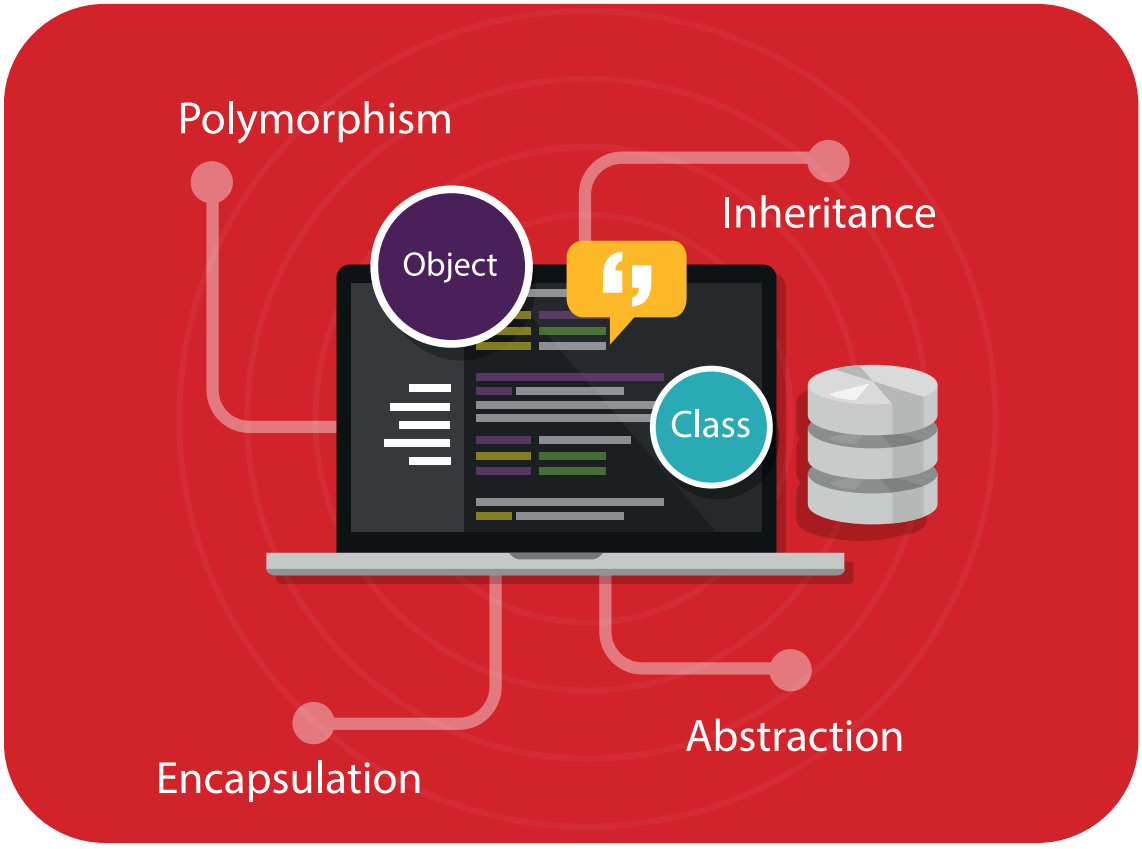
11. Aşağıdaki kodlar çalıştırıldığında program nasıl bir yol izler?

```
try
{
   int sayi;
   sayi = textBox1.Text;
}
catch
{
   MessageBox.Show("Hata bulundu!");
}
```

12. Üç farklı TextBox nesnesine girilen sayılardan hangisinin en büyük sayı olduğunu bulan programı yazınız.

3. ÖĞRENME BİRİMİ

SINIFLAR (CLASS)



KONULAR

3.1. SINIFLAR VE NESNELER
3.2. SINIF ÖZELLİKLERİ
3.3. METOT OLUŞTURMA VE ÇAĞIRMA
3.4. METOTLARI AŞIRI YÜKLEME
3.5. ERİŞİM BELİRLEYİCİLER
3.6. KAPSÜLLEME, KALITIM VE ÇOK BİÇİMLİLİK

ANAHTAR KELİMELER

Sınıf, nesne, alan, özellik, kapsülleme, metot, kalıtım, soyut sınıf, arayüz, çok biçimlilik, statik sınıf, isimsiz sınıf, mühürlü sınıf, parçalı sınıf, enums

NELER ÖĞRENECEKSİNİZ?

- Nesne tabanlı programlama mantığı
- Nesne tabanlı programlamanın temel prensipleri
- Sınıf ve nesneler oluşturma
- Sınıflarda alan, özellik ve metot ögeleri
- Erişim belirleyicileri
- Metotlar
- Değer ve referans kavramları
- Sınıflarda kalıtım özellikleri
- Soyut sınıf, arayüz ve çok biçimlilik kavramları
- Statik, isimsiz, mühürlü ve parçalı sınıflar
- Numaralandırma mantığı

HAZIRLIK ÇALIŞMALARI

1. Yapısal (prosedürel) programlamayı araştırınız.
2. Nesne tabanlı programlamanın en çok tercih edilen yazılım geliştirme yaklaşımı olmasının nedeni neler olabilir? Araştırınız.

3.1. NESNE TABANLI PROGRAMLAMAYA GİRİŞ

1960'lı yılların sonunda ortaya çıkan Nesne Tabanlı Programlama (NTP) kavramı; günümüzde yazılım geliştirmek, tasarlamak, analiz ve test etmek için kullanılır. NTP, öğrenilmesi gereken en önemli kavramlardan biridir. NTP bazı kaynaklarda "Nesne Yaklaşımlı Programlama", "Nesneye Yönelik Programlama", "Nesne Yönelimli Yaklaşım" olarak da kullanılır [(OOP) Object Oriented Programming - (OOA) Object Oriented Approach].

NTP, yazılımların boyutları ve karmaşıklığı arttığı için bazı yazılım gereksinimlerini karşılamak amacıyla doğmuştur. NTP, yazılan kodların bakımını ve aynı yazılım üzerinde birden fazla yazılımcının çalışmasını kolaylaştırır. Günümüzde geniş çaplı yazılım projelerinde yaygın olarak NTP kullanılır. NTP, fonksiyonel ve yapısal programlama tekniğine yeni bir bakış açısı getirmiştir. Günümüzde iyi bir yazılımcı olmak isteyen herkes NTP mantığını ve prensiplerini detaylı bir şekilde öğrenmelidir. Dünyada kullanılan programlama dillerinin birçoğu NTP'yi destekler.

3.1.1. NTP Öncesi

NTP öncesinde fonksiyonel programlama yaklaşımı mevcuttu. Fonksiyonel programlama, 1950'li yıllarda ilk üst düzey dillerin ortaya çıkışından bu yana kullanılır. Hem yapısal programlama hem de NTP genel amaçlı tüm programlama dillerinde uygulanır.

Yapısal programlama, büyük programları küçük parçalara bölerek çözümleme yöntemidir. Modüller ve alt programlar sıralı bir şekilde çalışır. Bu çalışma tarzı, ek bir programcılık yükü getirir ve kodların işleyişinin takibini zorlaştırır.

3.1.2. NTP Temel Prensipleri

NTP genel olarak dört prensip üzerine kurulmuştur. Herhangi bir programlama dilinin nesne tabanlı olduğundan söz edebilmek için asgari düzeyde şu prensipleri desteklemesi gerekir:

a) **Soyutlama (Abstraction):** Karmaşıklığın azaltılması anlamına gelir. Örneğin otomobillerde gaz pedalına basıldığında otomobil hızlanır ancak arka planda olan bitenler çoğu kişi için önemsizdir.

b) **Sarmalama veya Kapsülleme (Encapsulation):** Sadece istenilen bilgilerin dış dünyaya açılması, hassas veya özel bilgilerin gizlenmesi anlamına gelir. "Banka hesabına para yatır." komutu verildikten sonra T.C. Kimlik Numarası ve şifre bilgilerinin gizlenmesi buna örnek verilebilir.

c) **Kalıtım (Inheritance):** Var olan özelliklerin aktarılması anlamına gelir. Örneğin bütün kedi türlerinin dört ayaklı olması ortak bir özelliktir. Bir Van kedisi, kedilerin tüm özelliklerini taşırken ayrıca kendine has özellikleri de barındırır.

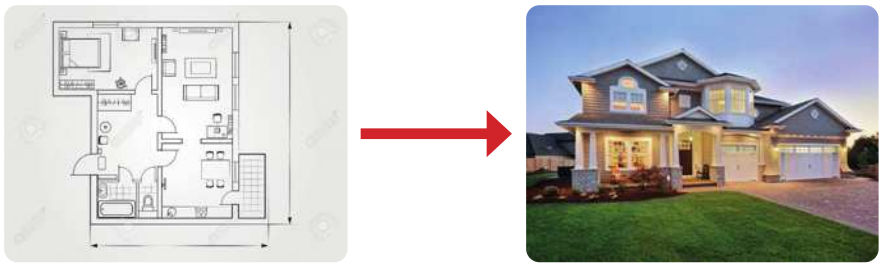
ç) Çok biçimlilik (Polymorphism): Farklı tiplere ait olan ortak özellikleri tanımlama işlemidir. Örneğin farklı hayvan türleri farklı sesler çıkarır zira "ses çıkarma" eylemi ortak bir özelliktir.

3.2. SINIFLAR VE NESNELER

Dünya ve çevre incelendiğinde her şeyin (cisimler, canlılar vb.) belirli **özelliklerinin** ve **işlevlerinin** olduğu görülür. Her öğrencinin bir numarası, adı soyadı, aldığı dersler gibi **özellikleri** ve okula gitme, sınava girme gibi **işlevleri** vardır. Benzer şekilde yine bir cep telefonunun rengi, boyutları, markası, adı gibi özellikleri ve çağrı başlatma, mesaj gönderme, uygulama açma gibi işlevleri bulunur.

NTP, dünyada var olan her şeyin yazılım içinde modellenmesini amaçlar. Sınıf (class), NTP'nin en önemli kavramıdır. Sınıf, nesnelerin özelliklerini ve işlevlerini (davranışlarını) tanımlamak için kullanılan bir taslaktır. Bu taslak aracılığıyla **nesneler** (objects) oluşturulur.

Görsel 3.1'de bir ev planı görülür. Programlamada bu ev planına sınıf, ev planından yola çıkılarak yapılan gerçek eve ise nesne adı verilebilir.



Görsel 3.1: Sınıf ve nesne ilişkisi

3.2.1. Sınıf Tanımlama

C#'ta sınıf tanımında class anahtar kelimesi kullanılır. Aşağıda en basit sınıf tanımı verilmiştir.

```
class SinifAdi
{
}
```

Yandaki örnekte bir dikdörtgen sınıfı tanımı yapılmış ve bu sınıf içinde birkaç farklı yapı kullanılmıştır. Bu yapıların ne olduğu ve ne amaçla yazıldığı alt başlıklarda anlatılmıştır.

```
class Dikdortgen
{
private int a, b;
public Dikdortgen(int a, int b)
{
this.a = a;
this.b = b;
}
public int AlanHesapla()
{
return a * b;
}
public int CevreHesapla()
{
return 2 * (a + b);
}
}
```

Bir dikdörtgenin iki kenar uzunluğu bilgisi bulunur. Ayrıca çevre ve alan bilgilerinin hesabı da söz konusudur. Sınıf tanımında a ve b değişkenleri dikdörtgenin kenar uzunluklarını saklamak için, **AlanHesapla()** ve **CevreHesapla()** işlevleri de dikdörtgenin alan ve çevre hesabının yapılması için tanımlanmıştır.

Not : C#'ta işlevleri gerçekleştiren kod bloklarına **metot** denir. Dolayısıyla bundan sonra işlev kelimesi yerine metot kelimesi kullanılacaktır.

Dikdörtgenle ilgili kod blokunda aşağıdaki gibi bir sınıf tanımlaması yapılmıştır.

```
class Dikdortgen
```

Sınıfın adı "Dikdortgen"dir. İsimlendirmelerde genel prensip olarak "ı, İ, ü, Ü, ö, Ö, ğ, Ğ, ş, Ş" gibi alfabemize özel harflerin kullanılmaması uygundur. Genel bir ifadeyle yazılacak olursa bir sınıf aşağıdaki gibi tanımlanır.

```
«Erişim belirleyici» class  «Sınıf adı»
{
}
```

Sıra Sizde

Dikdortgen sınıfına benzer şekilde Daire sınıfını oluşturunuz.

3.2.2. Nesne Oluşturma

Programlarda sınıfların kullanılabilmesi için bu sınıftan oluşturulan nesnelere (object) gereksinim duyulur. Bu türetme işlemine **örnek oluşturma (instance)** denir.

C#'ta önceden tanımlanan bir sınıftan **nesne** türetmek için **new** anahtar kelimesi kullanılır. Daha önceden oluşturulan Dikdortgen sınıfından bir nesne türetmek ve bu nesnenin ögelerini (özellikler ve metotlar) kullanmak için aşağıdaki gibi bir konsol uygulaması yazılabilir.

```
private static void Main(string[] args)
{
Dikdortgen d = new Dikdortgen(3, 4);
Console.WriteLine("Dikdörtgenin alanı: {0}", d.AlanHesapla());
Console.WriteLine("Dikdörtgenin çevresi: {0}", d.CevreHesapla());
}
```

Oluşturulan nesnenin adı "**d"**dir. Kenar uzunlukları 3 ve 4 olarak belirlenmiştir. Nesnenin ögelerine erişmek için nokta (.) karakterinin kullanıldığı görülür. Genel bir ifadeyle yazılacak olursa nesne aşağıdaki gibi tanımlanır.

```
«Sınıf adı» «Nesne adı» = new «Sınıf adı»(«Parametre listesi»);
```

Sıra Sizde

Dikdortgen nesnesine benzer şekilde yarıçapları farklı iki adet Daire nesnesi oluşturup metotlarını kullanınız.

3.3. KAPSÜLLEME, ALANLAR VE ÖZELLİKLER (ENCAPSULATION, FIELDS, PROPERTIES)

Erişim belirleyicileri ile NTP'nin temel prensiplerinden olan **kapsülleme,** alanlar ve özellikler aracılığıyla programlarda uygulanır. Kapsülleme, hassas veya özel bilgilerin gizlenmesi anlamına gelir.

3.4. ERİŞİM BELİRLEYİCİLER (ACCESS MODIFIERS)

.NET platformunda oluşturulan uygulamalarda güvenliği artırmak amacıyla sınıflara ve/veya sınıf içinde kullanılan ögelere erişimin kısıtlanması gerekir. Dolayısıyla koda dışarıdan erişimin sınırlarını belirlemek amacıyla erişim belirleyicileri kullanılır.

C# programlama dilinde kullanılan erişim belirleyicileri şunlardır:

public (Genel): Public olarak tanımlanan ögeler üzerinde herhangi bir kısıtlama yoktur. Her yerden erişilebilirdir.

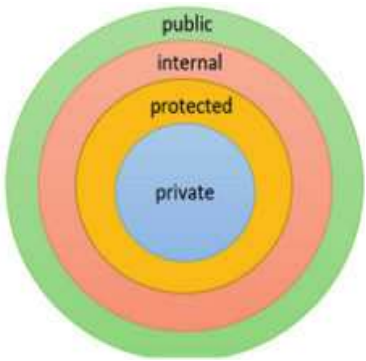
private (Gizli): En katı erişim belirleyicidir. Ögeler sadece tanımlandığı sınıf içinde erişilebilirdir. Bir başka deyişle ögeler sadece tanımlandığı küme parantezleri arasında kullanılabilir.

protected (Korunumlu): Ögeler, bulunduğu sınıf içinde ya da bu sınıftan türeyen diğer sınıflarda erişilebilirdir.

internal (Dâhilî): Internal olarak tanımlanan ögelere sadece aynı program içinden erişilebilir.

protected internal (Dâhilî+Korumalı): Ögeler hem protected hem de internal erişim belirleyicisine sahip olarak değerlendirilir. Türetilen sınıfın farklı program içinde olması sorun teşkil etmez.

Gizliden genele doğru erişim belirleyicileri Görsel 3.2'deki hiyerarşiye sahiptir.



Görsel 3.2: Erişim belirleyicileri hiyerarşisi

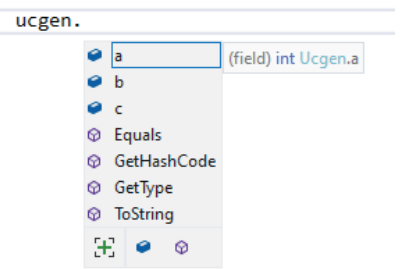
Not Bir ögeye herhangi bir erişim belirleyicisi tanımlaması yapılmazsa varsayılan olarak private kabul edilir.

3.5. ALANLAR (FIELDS)

Bir alan, sınıf içinde tanımlanmış herhangi türden (int, string vb.) bir değişkendir. Aşağıda Ucgen sınıfı ve bu sınıfa ait üç adet alan tanımlanmıştır.

```
public class Ucgen
{
    public int a;
    public int b;
    public int c;
  }
  internal class Program
  {
   private static void Main(string[] args)
   {
    Ucgen ucgen = new Ucgen();
    ucgen.a = 3;
    ucgen.b = 4;
    ucgen.c = 5;
    Console.WriteLine("Üçgenin a kenar uzunluğu: {0}", ucgen.a);
    Console.WriteLine("Üçgenin b kenar uzunluğu: {0}", ucgen.b);
    Console.WriteLine("Üçgenin c kenar uzunluğu: {0}", ucgen.c);
    }
}
```

Bu kod blokunda Ucgen sınıfından ucgen adında bir nesne türetilmiştir (C#'ın büyük / küçük harf duyarlı bir dil olduğu unutulmamalıdır. Dolayısıyla "U" ile "u" farklı karakterlerdir.).



Görsel 3.3: Sınıf ögelerine erişim

Nesnenin alanlarına erişim için Görsel 3.3'te görüldüğü üzere nokta (.) karakteri kullanılır. Kod editöründe nesnenin adı yazıldıktan sonra nokta (.) karakterine basıldığında sınıfa ait kullanılabilir ögelerin listesi gelir.

Sınıfa ait alanlar tanımlanırken başına "public" erişim belirleyicisi yazılmıştır. Bu erişim belirleyicisi, alan bilgisine sınıf dışından erişim için gereklidir. Alanlar yalnızca özel ve gizli kalması gereken değişkenler için kullanılmalıdır. Sınıf içinde tanımlanmış bir değişkenin başına yazılan "public" erişim belirleyicisi ile alanı dış dünyaya açmak uygun değildir. Bu şekilde yapıldığında değişkene değer atama ya da değişkenin değerinin okunması işlemlerinde kontrol mekanizması işletilemez. Ucgen sınıfına ait bir değişkene yandaki değer ataması kolaylıkla yapılabilir.

```
Ucgen ucgen = new Ucgen();
ucgen.a = -3;
ucgen.b = 0;
ucgen.c = -12345;}
```

Örnekte belirtilen kenar uzunluklarına sahip bir üçgeni çizmek mümkün değildir. Bu da programın doğru çalışmayacağı anlamına gelir. Bu durumun önüne geçmek için **özellikler** kullanılır.

Sıra Sizde

1. Öğrenci sınıfını ve alanlarını oluşturunuz.

2. Bilgisayar sınıfını ve alanlarını oluşturunuz.

3.6. ÖZELLİKLER (PROPERTIES)

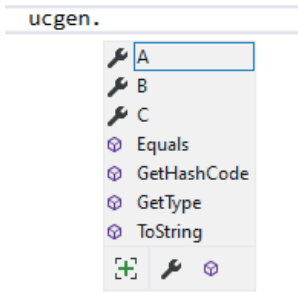
Bir değişkeni dış dünyaya açmak (diğer sınıflardan, programlardan vb. erişmek) için bu değişkene ait bir özellik eklenir. Bu sayede, NTP'nin temel prensiplerinden "kapsülleme" prensibi sınıfa uygulanır.

Ucgen sınıfının NTP prensiplerine daha uygun hazırlanmış örneği aşağıda verilmiştir.

```
public class Ucgen
{
    int a;
    int b;
    int c;

    public int A
    {
        get { return a; }
        set
        {
            if (value <= 0)
                Console.WriteLine("Hatalı bilgi");
            else
                a = value;
        }
    }
    public int B
    {
        get { return b; }
        set
        {
            if (value <= 0)
                Console.WriteLine("Hatalı bilgi");
            else
                b = value;
        }
    }
    public int C
    {
        get { return c; }
        set
        {
            if (value <= 0)
                Console.WriteLine("Hatalı bilgi");
        else
                c = value;
        }
    }
}}
```

Not Değişkenlerin başında erişim belirleyicisinin olmadığına dikkat edilmelidir.

Görsel 3.4'teki gibi sınıf adı yazılıp nokta (.) karakterine basıldığında alanlar değil, özellikler görüntülenir.



Görsel 3.4: Sınıf özelliklerine erişim

Not Alan ve özellik simgelerinin farklı olduğuna dikkat edilmelidir.

Get (almak, elde etmek) ve **set** (düzenlemek, ayarlamak) şeklinde iki ayrı alt metodu bulunur.

get Metodu: Bir değer döndürmek için kullanılır. Özelliklerin get metodunda **return** anahtar kelimesi kullanılarak "return…;" ile bir değerin döndürüleceği belirtilir.

set Metodu: Değişkene değer atama işlemleri için kullanılır. Burada görülen **value** anahtar kelimesi dışarıdan bu özelliğe gönderilen değeri temsil eder.

```
if (value <= 0)
   Console.WriteLine("Hatalı bilgi");
```

Yukarıdaki kod satırları ile dışarıdan alınan bilginin kontrolü gerçekleştirilir, sıfır veya negatif bir değer gönderildiğinde kullanıcıya hata mesajı gösterilir.

Özellikler kullanılarak Ucgen sınıfının alanlarına değer atamak için yandaki kod bloku yazılır.

```
Ucgen ucgen = new Ucgen();
ucgen.A = 3;
ucgen.B = 4;
ucgen.C = 5;
Console.WriteLine("Üçgenin a kenar uzunluğu: {0}", ucgen.A);
Console.WriteLine("Üçgenin b kenar uzunluğu: {0}", ucgen.B);
Console.WriteLine("Üçgenin c kenar uzunluğu: {0}", ucgen.C);
```

A kenarına negatif bir değer atanmak istendiğinde kullanıcıya bir uyarı mesajı gösterilir.

```
private static void Main(string[] args)
{
   Ucgen ucgen = new Ucgen();
   ucgen.A = -3;
}
// Ekran çıktısı:
Hatalı bilgi
```

Programlarda özellikler sıklıkla kullanıldığı için kısa yoldan dâhilî bir değişken ve bu değişkene ait özellik tanımlanabilir.

```
public class Ucgen
{
   public int A { get; set; }
   public int B { get; set; }
   public int C { get; set; }
}
```

Nesne oluştururken ilk değer ataması da kısa yoldan yapılabilir.

```
static void Main(string[] args)
{
    Ucgen u = new Ucgen
    {
        A = 3,
        B = 4,
        C = 5
    };
}
```

Yapıcı metotları aşırı yüklemeye gerek kalmadan doğrudan özelliklere değer ataması gerçekleştirilebilir. Özelliklerin programcılara sunduğu bazı avantajlar vardır. Her özellik;

- Sadece **get** metoduna,
- Sadece **set** metoduna,
- Hem **get** hem **set** metoduna sahip olabilir.

Sıra Sizde

1. Öğrenci sınıfını ve özelliklerini oluşturunuz.

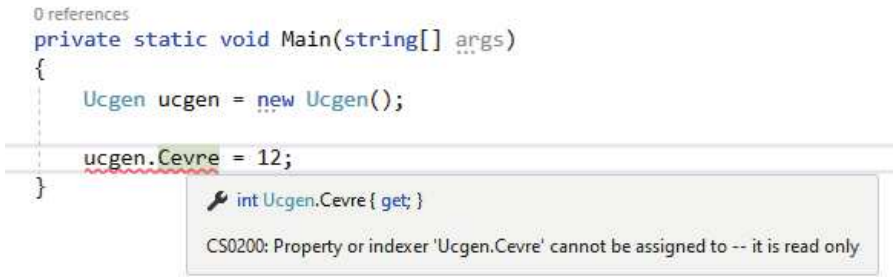
2. Bilgisayar sınıfını ve özelliklerini oluşturunuz.

3.6.1. Sadece Okunabilir Özellikler

Bir özellikte sadece **get** metodu kullanılırsa dışarıdan bu özelliğe değer ataması gerçekleştirilemez. Bu özellik = (eşittir) karakterinin solunda kullanılamaz. Bu özellik "sadece okunabilir" (readonly) bir özellik olur. Aşağıda sadece okunabilir bir özellik tanımı yapılmıştır.

```
public class Ucgen
{
    // ...
    public int Cevre
    {
        get
        {
            return a + b + c;
        }
    }
}
internal class Program
{
    private static void Main(string[] args)
    {
        Ucgen ucgen = new Ucgen();
        ucgen.A = 3;
        ucgen.B = 4;
        ucgen.C = 5;
        Console.WriteLine("Üçgenin çevresi: {0}", ucgen.Cevre);
    }
}
// Ekran çıktısı:
Üçgenin çevresi: 12
```

Cevre özelliğine değer atanmak istenirse bir hata oluşacaktır. Görsel 3.5'te görüldüğü gibi ucgen nesnesinin Cevre özelliğine değer atama işlemi gerçekleştirilemez.



Görsel 3.5: Salt okunur özelliğe değer atamaya çalışma

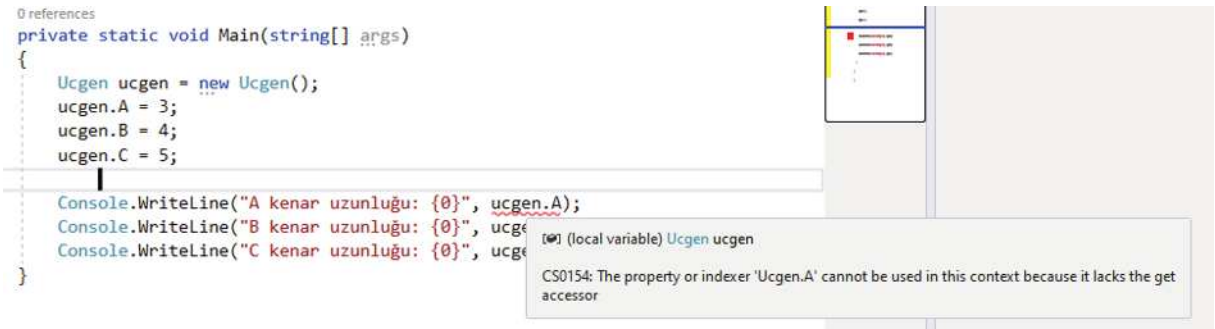
3.6.2. Sadece Yazılabilir Özellikler

Bir özellik sadece set metoduna sahipse bu özelliğe "sadece yazılabilir" özellik denir. Bu özellik = (eşittir) karakterinin sağında kullanılamaz. Dolayısıyla bu tür özellikler değer döndürmez, sadece dışarıdan değer alabilir. Daha önceden yazılan Ucgen sınıfının bir özelliği "sadece yazılabilir" özellik yapılmak istenirse bu özellik aşağıdaki gibi tanımlanmalıdır.

```
public class Ucgen
{
    int a;
    int b;
    int c;
    public int A
    {
        set
        {
            if (value <= 0)
                Console.WriteLine("Hatalı bilgi");
            else
                a = value;
        }
    }
    // ...
}
internal class Program
{
    private static void Main(string[] args)
    {
        Ucgen ucgen = new Ucgen();
        ucgen.A = 3;
        ucgen.B = 4;
        ucgen.C = 5;

      Console.WriteLine("A kenar uzunluğu: {0}", ucgen.A);
    }
}
```

Kod parçasında üçgenin a, b ve c kenarlarına özelliklerin set metodu üzerinden değer ataması yapılabildiği ancak ekrana yazdırma esnasında get metodu olmayan A özelliğinden bir değer döndürülemediği görülür.

Kod editöründe bu hata mesajı Görsel 3.6'daki gibi görülür.



Görsel 3.6: Sadece yazılabilir özelliği okumaya çalışma

3.7. METOTLAR (METHODS)

Bir metot, yalnızca çağrıldığında çalışan ve bir dizi ifade içeren kod blokudur. Yazılım dünyasında bir metot, sınıf içinde yapılacak işlerin veya operasyonların tanımlanmasını sağlar. Metotlara parametreler aracılığıyla ana programdan veriler gönderilebilir ve metotlar çalışmasını bitirdikten sonra ana programa değer döndürülebilir.

Genel olarak bir metot aşağıdaki şekilde tanımlanır.

```
«Erişim belirleyici» «Dönüş tipi» «Metodun adı» («Parametre listesi»)
{
}
```

Erişim Belirleyici: Metoda nerelerden erişilebileceğini tanımlar.

Dönüş Tipi: Metotlar değer döndürebilir. Dönüş tipi, metodun döndüreceği değerin tipini tanımlar (int, string vb.). Değer döndürmek için **return** anahtar kelimesi kullanılır. Metot değer döndürmeyecekse dönüş tipi olarak **void** anahtar kelimesi kullanılır.

Metot Adı: Metodun adını tanımlamada kullanılır.

Parametre Listesi: Parantez içinde verilen parametreler, bir metoda değer göndermek veya metottan değer almak için kullanılır. Parametrelerin türü, sayısı ve sırası parametre listesi olarak adlandırılır.

Bunlardan metodun adı ve parametre listesi **metodun imzası**, küme parantezi { } arasına yazılan kodlar da **metodun gövdesi** olarak adlandırılır. Metot imzaları, bir sınıf içinde her metotta farklı olmak zorundadır.

Metot adı + Parametre listesi = Metodun imzası

Aşağıda örnek metot tanımları verilmiştir.

```
public void ProgramiKapat() { ... }
public void DaireCiz(double x, double y, double cap) { ... }
public int KareKokHesapla(int sayi) { ... }
public void SMSGonder(string cepNo, string mesaj) { ... }
public decimal MaasHesapla(int gunSayisi, decimal gundelikUcret) { ... }
```

Bir metodu ana programdan çağırmak için metodun adı ile ( ) parantez açma ve kapatma karakterleri kullanılmalıdır.

```
nesneAdi.ProgramiKapat();
nesneAdi.DaireCiz(3.2, 2.4, 3);
int kareKok = nesneAdi.KareKokHesapla(9);
nesneAdi.SMSGonder("5051234567", "Merhaba, nasılsın?");
decimal maas = nesneAdi.MaasHesapla(24, 90.25M);
```

Verilen iki sayıdan büyük olanını bulan bir metot aşağıda yazılmıştır.

```
class SayiBulucu
{
   public int BuyukOlaniBul(int sayi1, int sayi2)
   {
      int sonuc;
      if (sayi1 < sayi2)
            sonuc = sayi2;
      else
            sonuc = sayi1;
      return sonuc;
   }
}
class Program
{
   static void Main(string[] args)
   {
      SayiBulucu sb = new SayiBulucu();
      int a = 100;
      int b = 25;
      int sonuc = sb.BuyukOlaniBul(a, b);
      Console.WriteLine("Büyük olan sayı: {0}", sonuc);
   }
}
```

SayiBulucu adlı sınıfın içindeki metotta;

- Erişim belirleyicisi **"public"**,
- Dönüş tipi **"int"**,
- Metot adı **"BuyukOlaniBul"**,
- Parametre listesi **"(int sayi1, int sayi2)"**,
- Metot imzası **"BuyukOlaniBul(int sayi1, int sayi2)"** olduğu söylenebilir.
- Metottan değer döndürmek için **return** anahtar kelimesi kullanılır.

Sıra Sizde

1. Parametresinde verilen sayının değeri tek ise true, çift ise false döndüren metodu yazınız.

2. Klavyeden 0 (sıfır) girilene kadar bu değerleri toplayıp döndüren metodu yazınız.

3.7.1. Varsayılan Değerli Parametreler (Optional Parameters)

Metot parametreleri tanımlanırken istendiğinde bunlara "varsayılan değerler" atanabilir. Metot çağrılırken bu parametrelere değer ataması yapılmazsa varsayılan değerler kullanılır.

```
class Sinifim
{
    public void Selamla(string ad = "Emre")
    {
        Console.WriteLine("Merhaba " + ad);
    }
}

class Program
{
    static void Main(string[] args)
    {
        Sinifim s = new Sinifim();
        s.Selamla();
        s.Selamla("Defne");
    }
}
// Ekran çıktısı:
Merhaba Emre
Merhaba Defne
```

Birden fazla parametre kullanıldığında varsayılan değere sahip parametreler, parametre listesinin en sonunda yer almalıdır. Aksi takdirde derleyici hatası oluşur (Görsel 3.7).

Görsel 3.7: Varsayılan parametrelerin hatalı kullanımı

3.7.2. İsimlendirilmiş Parametreler (Named Parameters)

Bir metot oluşturulurken tanımlanan parametrelere değer atamak için ana programda parametreler sırasıyla yazılmalıdır. İstenilirse parametre isimleri kullanılarak bu sıralamaya uyulmayabilir.

Yandaki kodda metot çağrılırken parametre adının ardından iki nokta (:) karakteri kullanılarak parametre sırasına uymadan atama yapıldığı görülür.

```
class SayiIslemleri
{
    public int Topla(int sayi1, int sayi2, int sayi3)
    {
        return sayi1 + sayi2 + sayi3;
    }
}
class Program
{
    static void Main(string[] args)
    {
        SayiIslemleri si = new SayiIslemleri();
        // int toplam = si.Topla(5, 10, 15);
      int toplam = si.Topla(sayi2: 10, sayi3: 15, sayi1: 5);
        Console.WriteLine("Toplam: {0}", toplam);
    }
}
```

3.7.3. Parametre Dizileri

Parametre, metodun parametre sayısının bilinmediği durumlarda **params** anahtar kelimesi ile tanımlanır.

params anahtar kelimesi ile bir parametre tanımlanacaksa bu parametre, metodun en son parametresi olmalıdır. Aksi takdirde derleyici hata verecektir. Ayrıca metotlarda sadece bir adet params tipinde parametre tanımlanabilir.

```
class SayiIslemleri
{
    public int Toplam(params int[] sayilar)
    {
        int toplam = 0;
        foreach (var s in sayilar)
        {
            toplam += s;
        }
        return toplam;
    }
}
class Program
{
    static void Main(string[] args)
    {
        SayiIslemleri si = new SayiIslemleri();
        Console.WriteLine("Toplam: {0}", si.Toplam(3));
        Console.WriteLine("Toplam: {0}", si.Toplam(3, 4, 5));
        Console.WriteLine("Toplam: {0}", si.Toplam(5, 1, 7, 3, 4, 5));
    }
}
// Ekran çıktısı:
Toplam: 3
Toplam: 12
Toplam: 25
```

3.7.4. Metodu Sonlandırma

Dönüş tipi **void** olan bir metodun çalıştırılması, istenildiği an sonlandırılabilir. Bunun için **return** anahtar kelimesi kullanılır.

```
class EkranIslem
{
    public void EkranaYaz(params int[] sayilar)
    {
        if (sayilar.Length == 0)
        {
            Console.WriteLine("Parametre olmadığı için metottan çıkılıyor.");
            return;
        }
```


Kod blokunun devamı sonraki sayfada

```
        Console.WriteLine("Parametreden gelen değerler:");
        foreach (var s in sayilar)
        {
            Console.WriteLine(s);
        }
    }
}
class Program
{
    static void Main(string[] args)
    {
        EkranIslem ei = new EkranIslem();
        ei.EkranaYaz(3, 4, 5);
        Console.WriteLine("========================");
        ei.EkranaYaz();
    }
}
// Ekran çıktısı:
Parametreden gelen değerler:
3
4
5
========================
Parametre olmadığı için metottan çıkılıyor.
```

Yukarıdaki kod blokunda **EkranaYaz** metodunun içinde **return** ifadesi ile metodun çalıştırılması sonlandırılmıştır.

Sıra Sizde

Yukarıdaki kod blokunda EkranaYaz metodunda verilen parametrelerden herhangi biri 0 (sıfır) ise programın çalışmasını sonlandıran değişikliği yapınız.

3.7.5. Metot Aşırı Yüklemeleri (Method Overloads)

Metodun adı aynı kalmak şartıyla parametre tipleri ve/veya sayısı değiştirilerek farklı metot imzaları oluşturulabilir. Bu durumda aynı isimli birden fazla metot oluşacaktır. Aşağıdaki sınıfta farklı türden parametreler alan **Topla** isimli metodun birden fazla kez oluşturulduğu görülür.

```
class ToplamaIslemi
{
    public int Topla(int a, int b)
    {
        Console.WriteLine("int parametreli metot çağrılıyor.");
        return a + b;
    }
```

Kod blokunun devamı sonraki sayfada

```
    public int Topla(params int[] sayilar)
    {
        Console.WriteLine("params parametreli metot çağrılıyor.");
        int toplam = 0;
        foreach (var s in sayilar)
        {
            toplam += s;
        }
        return toplam;
    }
    public double Topla(double a, double b)
    {
        Console.WriteLine("double parametreli metot çağrılıyor.");
         return a + b;
    }
    public string Topla(string a, string b)
    {
        Console.WriteLine("string parametreli metot çağrılıyor.");
         return a + b;
    }
}
class Program
{
    static void Main(string[] args)
    {
   ToplamaIslemi ti = new ToplamaIslemi();
        Console.WriteLine(ti.Topla(2, 5));
        Console.WriteLine("==============================");
        Console.WriteLine(ti.Topla(3.3, 5.1));
       Console.WriteLine("==============================");
         Console.WriteLine(ti.Topla("Sağlıcakla ", "kalınız."));
         Console.WriteLine("==============================");
         Console.WriteLine(ti.Topla(3, 8, 3, 7, 12, 33, 11, 4));
    }
}
// Ekran çıktısı:
int parametreli metot çağrılıyor.
7
==============================
double parametreli metot çağrılıyor.
8,4
==============================
string parametreli metot çağrılıyor.
Sağlıcakla kalınız.
==============================
params parametreli metot çağrılıyor.
81
```

Her bir Topla metodunun imzası farklı olduğu için derleyici hata vermez ve program başarılı bir şekilde çalıştırılır.

Not Görsel 3.8'deki kod editöründe metot adından sonra kullanılabilir parametre tipleri listelenmiştir.

Görsel 3.8: Metot aşırı yüklenmelerini kullanma

Burada Topla metodunun dört farklı aşırı yüklenmiş hâli olduğu belirtilir. Klavyeden yukarı ve aşağı tuşları ile parametre tipleri incelenebilir.

Sıra Sizde

Yaş hesaplayan bir metodu DateTime (doğum tarihi) ve int (doğum yılı) parametrelerini alacak şekilde aşırı yükleyerek gerçekleştiriniz.

3.8. YAPICI VE YIKICI METOTLAR

Nesneler oluşturulduğunda ve yok edilme anında otomatik olarak çalıştırılan metotlar vardır. Nesneler oluşturulurken otomatik olarak çalıştırılan metotlara **yapıcı metot** (constructor), nesnelerin yok edildiği anda otomatik olarak çalıştırılan metotlara **yıkıcı metot** (destructors) denir.

3.8.1. Yapıcı Metotlar (Constructors)

Yapıcı metotlar, nesnelerin ilk oluşturulduğu anda otomatik olarak çalıştırılır. Yapıcı metotlar genellikle sınıf içinde tanımlanan yerel değişkenlerin ilk değerlerini düzenlemek için kullanılır.

Bir metodun yapıcı metot olabilmesi için şu şartları taşıması gerekir:

- Metot adı, sınıfın adı ile aynı olmalıdır.
- Geri dönüş tipi olmamalıdır (void ya da int gibi).
- Nesneleri oluşturmak için **new** operatörü kullanıldığı anda yapıcı metotlar otomatik olarak çalıştırılır. Nesne oluşturulduktan sonra yapıcı metotlar bir daha çağrılamaz.

Not

- Sınıf içinde bir yapıcı metot tanımlanmamışsa derleyici arka planda boş bir varsayılan yapıcı metot oluşturur.
- Yapıcı metotlar bazı kaynaklarda "kurucu metot" veya "oluşturucu metot" olarak da geçer.

Aşağıda Kisi sınıfında tanımlanan iki adet yerel değişkene yapıcı metot içinde değer ataması yapılmıştır.

```
  class Kisi
{
    int yas;
    string ad;
    public Kisi()
    {
        yas = 19;
        ad = "Ahmet";
        Console.WriteLine("Yapıcı metot çalıştı.");
    }
    public int Yas
    {
        get
        {
            return yas;
        }
    }

    public string Ad
    {
        get
        {
            return ad;
        }
    }

}
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Program başladı.");
        Kisi k = new Kisi();
        Console.WriteLine("Adı: {0}, Yaşı: {1}", k.Ad, k.Yas);
        Console.WriteLine("Program bitti.");
    }
}
```

```
// Ekran çıktısı:
Program başladı.
Yapıcı metot çalıştı.
Adı: Ahmet, Yaşı: 19
Program bitti.
```

Kod parçasında **new Kisi()** komutu işletildiği anda yapıcı metodun çalıştırıldığı görülür. Yapıcı metotları da aşırı yüklemek mümkündür.

Kisi sınıfı şu şekilde de tanımlanabilir:

```
  class Kisi
{
   int yas = 0;
   string ad = "";

   public Kisi()
   {
      yas = 19;
      ad = "Ahmet";
      Console.WriteLine("Yapıcı metot çalıştı.");
   }
   public Kisi(int yas)
   {
      this.yas = yas;
      ad = "Ahmet";
      Console.WriteLine("int parametreli yapıcı metot çalıştı.");
   }
   public Kisi(string ad)
   {
      yas = 19;
      this.ad = ad;
      Console.WriteLine("string parametreli yapıcı metot çalıştı.");
   }
   public Kisi(int yas, string ad)
   {
      this.yas = yas;
      this.ad = ad;
      Console.WriteLine("İki parametreli yapıcı metot çalıştı.");
   }
}
class Program
{
   static void Main(string[] args)
   {
      Console.WriteLine("Program başladı.");
      Kisi k1 = new Kisi();
      Kisi k2 = new Kisi(23);
      Kisi k3 = new Kisi("Filiz");
      Kisi k4 = new Kisi(25, "Süleyman");
      Console.WriteLine("Program bitti.");
   }
}
```

```
// Ekran çıktısı:
Program başladı.
Yapıcı metot çalıştı.
Adı: Ahmet, Yaşı: 19
Program bitti.    public int Topla(params int[] sayilar)
```

Yapıcı metot içinde kullanılan **this** anahtar kelimesi, bu sınıftan türeyen nesneyi temsil eder. Dolayısıyla parametre adı ile sınıf değişkeninin adlarının aynı olması durumunda **this** anahtar kelimesi ile bu sınıftan türetilen nesnenin ilgili değişkenini kullanmak mümkündür.

```
public Kisi(int yas)
{
   this.yas = yas;
}
```

Parametreden gelen **yas** bilgisi, **"this.yas"** ifadesi ile bu sınıftan türetilen nesnenin **yas** alanına atanır.

Sıra Sizde

1. Daire sınıfını yapıcı metodu ile beraber tanımlayınız (Yarıçap değerini almalıdır.).

2. Ev sınıfını yapıcı metodu ile beraber tanımlayınız (Oda sayısı ve m^2 değerlerini almalıdır.).

3.8.2. Yıkıcı Metotlar (Destructors)

Nesne hafızadan atıldığı anda otomatik olarak çalışan yıkıcı metotlar, tıpkı yapıcı metotlar gibi özel metotlardır ve şu şartları taşımalıdır:

- Metot adı, sınıfın adı ile aynı olmalıdır.
- Metot adının başında ~ (Tilde) karakteri olmalıdır.
- Bir sınıfın yalnızca bir tane yıkıcı metodu olabilir.
- Yıkıcı metotlar aşırı yüklenemez.
- Yıkıcı metotlar parametre alamaz.
- Yıkıcı metotların erişim belirleyicisi olamaz.

Programcının yıkıcı metot üzerinde bir kontrolü bulunmaz. Yıkıcı metotlar genellikle sınıf içinde kullanılan kaynakların (veri tabanı, dosya vb.) kapatılması ve hafızadan atılması amacıyla .**NET Framework** içindeki **Garbage Collector** (Çöp Toplayıcısı) tarafından gerekli görüldüğü zaman çalıştırılır.

Yıkıcı metoda sahip bir sınıf örneği aşağıda verilmiştir.

```
class Otomobil
{
   string marka = "";
   string renk = "";
   public Otomobil()
   {
      marka = "TOGG";
      renk = "kırmızı";
      Console.WriteLine("Yapıcı metot çalıştı.");
   }
   ~Otomobil()
```

Kod blokunun devamı sonraki sayfada

```
    {
        Console.WriteLine("Nesne hafızadan atıldı.");
    }
}
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Program başladı.");
        Otomobil oto = new Otomobil();
    Console.WriteLine("Program bitti.");
    }
}
```

```
// Ekran çıktısı:
Program başladı.
Yapıcı metot çalıştı.
Program bitti.
Nesne hafızadan atıldı.
```

Sıra Sizde

Yıkıcı metotların kullanım yerlerini araştırınız, edindiğiniz bilgileri sınıf arkadaşlarınızla paylaşınız.

3.9. DEĞER VE REFERANS TİPLER

.NET platformunda hafıza yönetiminin nasıl işlediğinin bilinmesi, programcılar için oldukça önemlidir. .NET'te hafıza, **yığın** (stack) ve **öbek** (heap) olmak üzere iki bölgeye ayrılmıştır.

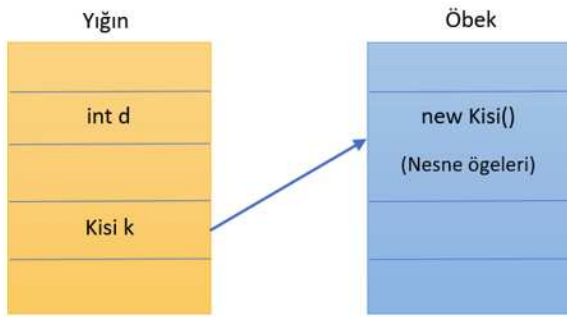
Değişkenler, veri tipine göre atanan değerleri taşıyan veri tutuculardır. .NET platformunda kullanılan her bir veri tipi **değer tipli** ve **referans tipli** olarak ikiye ayrılır. Bu veri tiplerinden değer veri tipleri, hafızanın yığın bölgesinde; referans veri tipleri de hafızanın öbek bölgesinde tutulur.

Değer Tipleri: int, long, float, double, decimal, char, bool, byte, short, struct, enum

Referans Tipleri: string, object, class, interface, array, delegate

Not Bunlardan **string** veri tipi teorikte referans tipli olmasına rağmen program içinde değer tipli olarak işlem görür.

Değer tipleri, veriyi bizzat barındıran türlerdir. Referans tipleri ise veri yerine verinin bellekteki adresini tutan türlerdir (Görsel 3.9).



Görsel 3.9: Yığın ve öbek ilişkisi

Değer tiplerinden biri kullanılarak bir değişken tanımlandığında değişkenin değeri yığın bellek bölgesinde tutulur. Referans tipte bir değişken tanımlandığında ise değişkenin değeri öbek hafıza bölgesinde tutulur ve bu bölgenin adresini tutan bilgi de yığında saklanır. Yığında öbek bölgesini işaret eden bir işaretçi (pointer) oluşturulur.

Değer ve referans tiplerin çalışma mantığı aşağıda verilmiştir.

```
class SayiTutucu
{
    public int A { get; set; }
}
class Program
{
    static void Main(string[] args)
    {
        int sayi1 = 10;
        int sayi2 = sayi1;
        sayi2 = 50;
        Console.WriteLine("sayi1 = {0}", sayi1);
        Console.WriteLine("sayi2 = {0}", sayi2);
        Console.WriteLine("======================");
        SayiTutucu st1 = new SayiTutucu();
        st1.A = 10;
        SayiTutucu st2 = st1;
        st2.A = 50;
        Console.WriteLine("st1.A değeri: {0}", st1.A);
        Console.WriteLine("st2.A değeri: {0}", st2.A);
    }
}
```

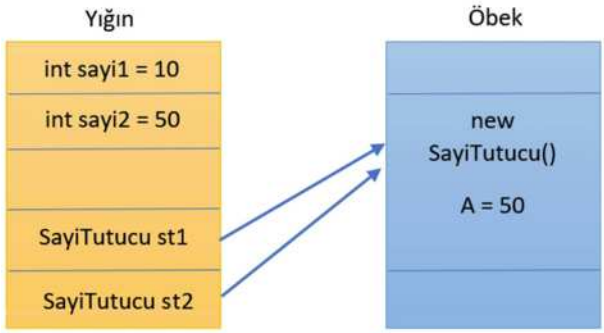
```
// Ekran çıktısı:
sayi1 = 10
sayi2 = 50
======================
st1.A değeri: 50
st2.A değeri: 50
```

Main metodu çalıştırıldığında sırasıyla şu işlemler gerçekleşir:

- **int sayi1 = 10;** => sayi1 için yığında alan ayrılır ve buraya 10 değeri yazılır.
- **int sayi2 = sayi1;** => sayi2 için yığında alan ayrılır ve bu alana sayi1'in değeri 10 yazılır.
- **sayi2 = 50;** => sayi2 için ayrılan alana 50 değeri yazılır. sayi1 değişkeninin değeri değişmez.
- **SayiTutucu st1 = new SayiTutucu();** => Oluşturulan nesne için öbekte bir alan ayrılır ve nesnenin ögeleri burada saklanır. Bu alanın adresi st1 değişkeninde tutulur. Bu değişken için de yığında bir yer ayrılır.
- **st1.A = 10;** => st1'in gösterdiği öbek alanındaki nesnenin A özelliğinin değeri 10 olarak güncellenir.

- **SayiTutucu st2 = st1;** => st2 için yığında ayrı bir yer ayrılır ve buraya st1 değişkeninin tuttuğu adres bilgisi yazılır. Dolayısıyla st1 ve st2 aynı öbek alanının adresini tutar.
- **st2.A = 50;** => st2'nin gösterdiği öbek alanındaki nesnenin A özelliğinin değeri 50 olarak güncellenir.

st1 ve st2, aynı öbek alanının adresini tuttuğu için hangi değişken üzerinden olduğu fark etmeksizin aynı nesnenin A özelliğinin değeri ekrana yazdırılacaktır (Görsel 3.10).



Görsel 3.10: Aynı referansa sahip nesneler

3.9.1. Metotlarda ref ve out Kullanımı

Değer tipli değişkenler metotlara parametre olarak gönderildiğinde bu değişkenin değeri için yığında farklı bir bellek alanı ayrılır. Dolayısıyla metot içinde bu değişkenin değeri değiştirilse bile değişiklik ana programdan gönderilen değişkeni etkilemez. Aşağıdaki kod parçasında bu durum verilmiştir.

```
class Matematik
{
    public void Artir(int x)
    {
        x++;
    }
}
class Program
{
    static void Main(string[] args)
    {
        Matematik m = new Matematik();
        int a = 100;
        m.Artir(a);
        Console.WriteLine("a değeri = {0}", a);
    }
}
// Ekran çıktısı:
a değeri = 100
```

Artir metodunun içinde x değişkeninin değeri 1 artırılmasına rağmen bu değişikliğin ana metot içinde kullanılan a değişkeninin değerine bir etkisi olmaz çünkü ana metot içindeki a değişkeni ile Artir metodunda kullanılan x değişkeni için yığında farklı hafıza alanları ayrılır ve bunlar birbirlerinden tamamen ayrı iki değişken olarak düşünülmelidir.

Parametre olarak gönderilen değişkenin değeri Artir metodunun içinde değiştirilmek istenirse bu durumda **ref** veya **out** anahtar kelimeleri kullanılmalıdır.

Bir önceki kod parçacığı ref anahtar kelimesi kullanılarak tekrar yazılırsa ana metot içindeki değişkenin değerinin değiştiği görülür.

```
class Matematik
{
    public void Artir(int x)
    {
        x++;
    }
}
class Program
{
    static void Main(string[] args)
    {
        Matematik m = new Matematik();
        int a = 100;
        m.Artir(a);
        Console.WriteLine("a değeri = {0}", a);
    }
}
// Ekran çıktısı:
a değeri = 100
```

Yandaki kodda Artir metodu yazılırken ve bu metot çağrılırken ref anahtar kelimesinin kullanıldığına dikkat edilmelidir. ref veya out anahtar kelimeleri ile metoda parametre gönderildiğinde aslında değişkenin değeri değil, değişkenin yığında bulunduğu hafıza adresi gönderilir. Dolayısıyla metot içinde değişken üzerinde yapılan değişiklikler aslında ana metotta kullanılan değişken üzerinde gerçekleşir.

out anahtar kelimesinin kullanımı, ref anahtar kelimesinin kullanımından biraz farklıdır. ref kullanılmak istendiğinde parametre olarak göndermeden önce değişkene mutlaka bir değer ataması yapılır. out kullanıldığında ise değişkenin değeri artırma veya azaltma gibi bir işleme tabi tutulmadan önce değişkene mutlaka değer ataması yapılır.

Kod parçası out anahtar kelimesi ile tekrar yazılmıştır.

```
class Matematik
{
    public void Artir(out int x)
    {
        x = 123;
    }
}
class Program
{
    static void Main(string[] args)
    {
        Matematik m = new Matematik();
        int a;
        m.Artir(out a);
        Console.WriteLine("a değeri = {0}", a);
    }
}
// Ekran çıktısı:
a değeri = 123
```

Tablo 3.1'de ref ve out anahtar kelimeleri arasındaki farklar listelenmiştir.

Tablo 3.1: Aynı Referansa Sahip Nesneler

ref	out
Metodu tanımlarken parametrenin önüne "ref" yazılmalıdır.	Metodu tanımlarken parametrenin önüne "out" yazılmalıdır.
Metodu çağırırken değişkenin önüne "ref" yazılmalıdır.	Metodu çağırırken değişkenin önüne "out" yazılmalıdır.
Metoda göndermeden önce değişken başlangıç değeri almak zorundadır.	Metoda göndermeden önce değişken başlangıç değeri almak zorunda değildir.
Metot içinde istenildiği gibi kullanılabilir.	Metot içinde mutlaka bir değer ataması gerçekleştirilmelidir.

Sıra Sizde

Siz de ref veya out kullanarak ad, soyad ve yaş bilgilerini döndüren metodu yazınız.

3.10. KALITIM (INHERITANCE)

Kalıtım, NTP'deki en önemli kavramlardan biridir ve bir sınıfın özelliklerinin farklı sınıflar tarafından da kullanılabilmesini sağlar. Buna **miras alma** da denir. Bu durumda miras alınan sınıfa **üst** veya **temel sınıf** (parent), miras alan sınıfa da **türetilmiş sınıf** (derived) denir. C#'ta bir sınıf sadece bir sınıftan türetilebilir.

Bu durumun tek istisnası **Object** sınıfıdır. Bir sınıf, başka bir sınıftan türesin veya türemesin, varsayılan olarak **Object** sınıfından türer. Buna **örtük devralma** denir. Boş bir sınıftan oluşturulan nesnelerin sahip olduğu metotlar **Object** sınıfından gelir (Görsel 3.11).

```
1 reference
class SinifAdi
{

}
0 references
internal class Program
{
    0 references
    private static void Main(string[] args)
    {
        SinifAdi sa = new SinifAdi();
        sa.
    }
}
```

Equals
GetHashCode
GetType
ToString

Görsel 3.11: Object sınıfından miras alınan ögeler

Üst sınıf ile türetilmiş sınıf arasında bir üst / alt ilişkisi vardır. Türetilmiş sınıf, üst sınıf ögelerine erişebilir ancak bu durumun tersi doğru değildir.

Bir sınıfı bir başka sınıftan türetmek için iki nokta (:) karakteri kullanılır.

```
«Erişim belirleyici» class «Sınıf adı» : «Üst sınıf adı»
{
}
```

OkulPersoneli sınıfı ve bu sınıftan türetilen bir **Ogretmen** sınıfı aşağıda tanımlanmıştır.

```
public class OkulPersoneli
{
    public string Ad { get; set; }
    public string Soyad { get; set; }
}
public class Ogretmen : OkulPersoneli
{
    public string Brans { get; set; }
}
class Program
{
    static void Main(string[] args)
    {
        Ogretmen ogr = new Ogretmen
        {
            Ad = "Ahmet",
            Soyad = "Öz",
            Brans = "Matematik"
        };
        // ..
    }
}
```

Kod parçasında Ad ve Soyad özellikleri Ogretmen sınıfına aktarılmıştır. Ogretmen sınıfı, Ad ve Soyad bilgilerini OkulPersoneli sınıfından miras almıştır. Miras alma işlemlerinde "Her ... bir ... dır." mantığı vardır. Kod parçası için şu ifade kurulabilir: "Her öğretmen bir okul personelidir."

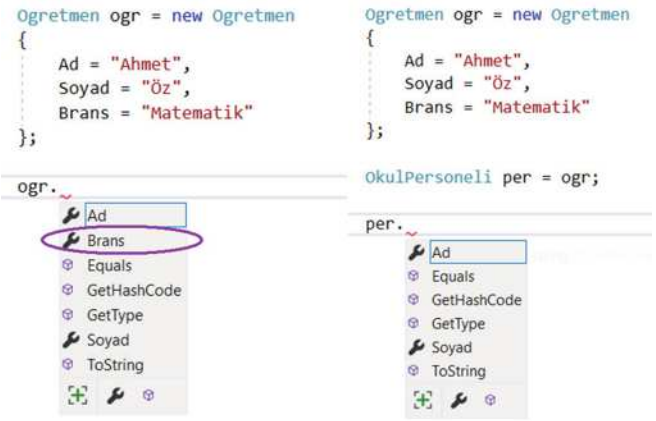
Her öğretmen bir okul personeli olduğuna göre aşağıdaki kod satırı hata vermeyecektir.

```
Ogretmen ogr = new Ogretmen
{
    Ad = "Ahmet",
    Soyad = "Öz",
    Brans = "Matematik"
};
OkulPersoneli per = ogr; // !!!
Console.WriteLine(per.Ad);

// Ekran çıktısı:
Ahmet
```

Hafızanın öbek bölgesinde Ogretmen nesnesi bulunmasına rağmen per değişkeni üzerinden sadece Ad ve Soyad özelliklerine erişim mümkündür (Görsel 3.12).

Sıra Sizde

Çevrenizdeki nesneleri kalıtım açısından inceleyiniz ve en az üç tane kalıtım örneği gerçekleştiriniz.



Görsel 3.12: Referansın tipine bağlı nesneye erişim

3.10.1. Hiyerarşik Kalıtım

Bir sınıf, türetilmiş sınıflardan kalıtım yoluyla aynı şekilde türetilebilir. Bir anlamda hiyerarşik kalıtım mümkündür.

```
public class Canli
{
  //...
}
public class Hayvan : Canli
{
  //...
}
public class Kopek : Hayvan
{
  //...
}
public class Kangal : Kopek
{
  //...
}
```

Sınıf tanımlamaları geçerlidir ve hiyerarşik kalıtıma bir örnektir. "Her ... bir ... dır." mantığı her bir sınıf için geçerlidir.

3.10.2. new Operatörüyle Metot Gölgeleme (Shadowing)

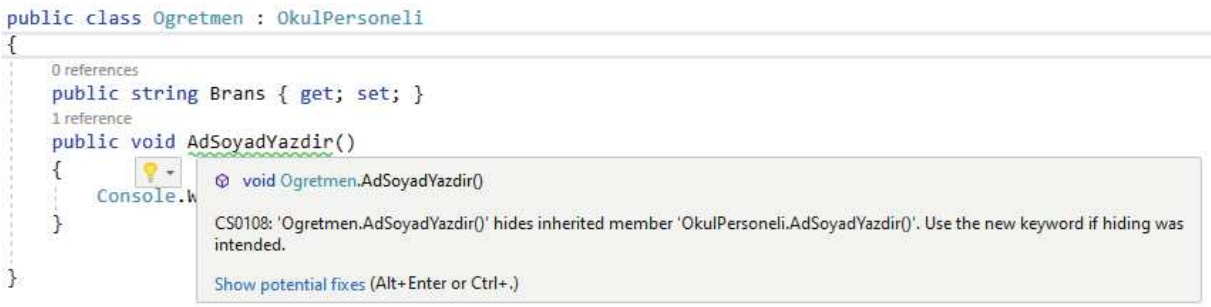
Bir sınıftan başka bir sınıf türetildiğinde özel bir durum ortaya çıkar. Üst sınıfta bulunan bir metot, alt sınıfta da tanımlanmışsa derleyici bir "uyarı" verir. Bu durumda üst sınıfta yer alan metot gölgelenerek erişilemez duruma gelir.

```
public class OkulPersoneli
{
   public string Ad { get; set; }
   public string Soyad { get; set; }
   public void AdSoyadYazdir()
   {
      Console.WriteLine("Benim adım soyadım : " + Ad + " " + Soyad);
   }
}
```

Kod blokunun devamı sonraki sayfada

```
public class Ogretmen : OkulPersoneli
{
    public string Brans { get; set; }

    public void AdSoyadYazdir()
    {
        Console.WriteLine("Benim adım soyadım : " + Ad + " " + Soyad);
    }
}
```

Kod parçasında **Ogretmen** sınıfındaki **AdSoyadYazdir()** metodu, üst sınıftaki aynı isimli metodu gölgeler. Hem üst sınıfta hem de türetilmiş sınıfta aynı isimli metot bulunduğu için derleyici, alt sınıftaki metodun **new** operatörü ile tanımlanması gerektiğini belirten bir "uyarı" (hata değil) verir (Görsel 3.13).



Görsel 3.13: Derleyici uyarı mesajı

Bu durum bir hata olmadığı için program çalışır ancak uyarı mesajını ortadan kaldırmak için bu metodun kasıtlı olarak yazıldığını derleyiciye bildirmek gerekir. Bunun için **new** anahtar kelimesi kullanılır.

```
public class Ogretmen : OkulPersoneli
{
    public string Brans { get; set; }
    public new void AdSoyadYazdir()
    {
        Console.WriteLine("Benim adım soyadım : " + Ad + " " + Soyad);
    }
}
```

Bu durumda derleyici herhangi bir uyarı ya da hata mesajı vermez ancak bu kullanımın NTP prensiplerine uygun olduğu söylenemez. Bu gibi durumlarda "sanal metot"ların kullanılması uygundur.

3.10.3. Sanal Metotlar (Virtual Methods)

Temel sınıftan türetilmiş alt sınıflara aktarılan metotlar her zaman olduğu gibi kullanılmayabilir. İstenilen metotlar alt sınıflarda tekrardan yazılabilir. Böyle bir durumda bu metotlar, üst sınıfta **virtual** (sanal), alt sınıflarda da **override** (geçersiz kılma veya ezme) anahtar kelimeleri kullanılarak tanımlanmalıdır. Sanal olarak tanımlanan metotlar, alt sınıflarda geçersiz kılınmak zorunda değildir. Sanal metotlar geçersiz kılınmazsa metodun kendi çağrılır. Sanal metotlar geçersiz kılınırsa alt sınıfın metodu çağrılır.

```
public class Sekil
{
    public const double pi = 3.14;
    protected double x, y;
    public Sekil()
    {
    }
    public Sekil(double x, double y)
    {
        this.x = x;
       this.y = y;
    }
public virtual double AlanHesapla()
    {
        return x * y;
    }
    public virtual void BilgiYazdir()
    {
        Console.WriteLine("x= " + x + " ve y= " + y);
    }
}
public class Daire : Sekil
{
    public Daire(double r) : base(r, 0)
    {
    }
    public override double AlanHesapla()
    {
        return pi * x * x;
    }
}
class Program
{
    static void Main(string[] args)
    {
        Sekil s = new Sekil(3, 4);
        s.BilgiYazdir();
        Console.WriteLine("Şekil alanı: " + s.AlanHesapla());
        Console.WriteLine("======================");
        Daire d = new Daire(1.3);
        d.BilgiYazdir();
        Console.WriteLine("Daire alanı: {0:N2}", d.AlanHesapla());
    }
}
```

```
// Ekran çıktısı:
x= 3 ve y = 4
Şekil alanı: 12
=======================
x= 1,3 ve y= 0
Daire alanı: 5,31
```

Daire sınıfının alan hesabı, Sekil sınıfının alan hesabından farklıdır. Programda da görüldüğü gibi AlanHesapla() metodu alt sınıfta geçersiz kılınmış ancak BilgiYazdir() metodu alt sınıfta geçersiz kılınmamıştır.

Not

```
public Daire(double r) : base(r, 0)
        {
        }
```

Kod blokunda Daire sınıfının yapıcı metodu tanımlanırken nesne oluşturulduğu anda aynı zamanda üst sınıfın yapıcı metodunun da çağrılması sağlanmıştır. Bunun için base anahtar kelimesi kullanılır. this anahtar kelimesi sınıfı, base anahtar kelimesi ise üst sınıfı temsil eder.

Not

Sınıf içinden üst sınıfın ögelerine **base** anahtar kelimesi ile erişim mümkündür. Bu örnek için **"base.x"** ifadesi ile Sekil sınıfının x alanına erişilebilir.

Sıra Sizde

Kitap, dergi ve ansiklopediler için bir üst sınıf yazarak Oku() sanal metotlarını ihtiva eden sınıflar tanımlayınız.

3.11. SOYUT SINIFLAR (ABSTRACT CLASSES)

NTP'nin bir diğer önemli kavramı da soyutlamadır. Soyutlama genellikle ortak özellikleri olan sınıfları bir çatı altında toplamak için kullanılır. Soyut sınıfların klasik sınıflardan en önemli farkı, new anahtar kelimesi ile nesnelerinin oluşturulamamasıdır.

Soyut sınıflar **abstract** anahtar kelimesi ile tanımlanmalı ve en az bir tane **abstract** ile tanımlanmış metodu olmalıdır. Bu metodun sadece imzası bulunur, gövdesi bulunmaz. Ayrıca soyut olarak tanımlanmış metotlar, bu sınıftan türeyen alt sınıflarda mutlaka geçersiz kılınmalıdır (override).

Not

Sanal metotlar geçersiz kılınmak zorunda değildir. Buna karşın soyut metotlar mutlaka geçersiz kılınmalıdır.

```
public abstract class MotorluArac
{
    public int MotorHacmi { get; set; }
    public int ModelYili { get; set; }

    public abstract void Calis();
    public abstract void Dur();
}
public class Otomobil : MotorluArac
{
    public bool OtomatikVites { get; set; }

    public override void Calis()
    {
        Console.WriteLine("Otomobil çalıştı.");
    }
    public override void Dur()
    {
        Console.WriteLine("Otomobil durdu.");
    }
}
class Program
{
    static void Main(string[] args)
{
        // ** Alttaki satır hata verir.
        // ** Sanal sınıflardan nesne türetilemez.
        // MotorluArac ma = new MotorluArac();
    Otomobil oto = new Otomobil
        {
            ModelYili = 2020,
            MotorHacmi = 1600,
            OtomatikVites = true
        };
        oto.Calis();
        oto.Dur();
    }
}
// Ekran çıktısı:
Otomobil çalıştı.
Otomobil durdu.
```

Her motorlu araçta bulunan ortak özellikler MotorluAraclar sınıfında, otomobillere özgü özellikler de Otomobil sınıfında tanımlanmıştır.

MotorluAraclar sınıfındaki iki soyut metot, Otomobil sınıfında geçersiz kılınmıştır (Geçersiz kılınmak zorundadır.).

Sanal metotların sadece imzaları vardır, gövdeleri yoktur.

```
public abstract void Calis();
public abstract void Dur();
```

"Her ... bir ... dır." mantığı burada da geçerlidir. "Her otomobil bir motorlu araçtır." ifadesi doğru olduğuna göre aşağıdaki kod hata vermeyecektir.

```
Otomobil oto = new Otomobil
{
    ModelYili = 2020,
    MotorHacmi = 1600,
    OtomatikVites = true
};
 MotorluArac ma = oto;
ma.Calis();
ma.Dur();
```

Sıra Sizde

Kitap, dergi ve ansiklopediler için bir soyut üst sınıf yazarak Oku() metotlarını ihtiva etmek zorunda olan sınıfları tanımlayınız.

3.12. ARAYÜZLER (INTERFACES)

NTP'de soyutlamanın bir başka yolu da **arayüzler** (interfaces) aracılığıyla mümkündür. Bir arayüz, tüm ögeleri soyut olan bir sınıfa benzetilebilir ancak burada arayüzlerin ve sınıfların farklı kavramlar olması önemli bir husustur. Bir sınıf sadece bir sınıftan türetilebiliyorken birden fazla arayüzden türetilebilir.

Arayüzün içinde tanımlanan metotların sadece imzaları bulunur, gövdeleri bulunmaz. Ayrıca arayüzde bulunan tüm metotlar varsayılan olarak soyuttur (abstract) ve genel (public) erişim belirleyicisine sahiptir.

Arayüz bir sınıf türü olmadığından içinde kod bloku bulunamaz. Arayüzde tanımlanan ögeler, kendinden türetilen sınıfta mutlaka uygulanmak (implement) zorundadır.

Not

- Soyut sınıflarda "geçersiz kılma" (override) kavramı, arayüzler için de "uygulamak" (implement) kavramı kullanılır.
- Arayüz isimlerinin sınıf olmadığını belirtmek için "I" (büyük ı) harfi ile başlatılması gelenektir.

Türetildiği arayüzleri uygulayan bir sınıf tanımı aşağıda verilmiştir.

```
interface IHayvan
{
    void SesCikar();
}
interface IBeslen
{
    void Beslen();
}
public class Kedi : IHayvan, IBeslen
{
    public void SesCikar()
    {
        Console.WriteLine("Kedi: miyav");
    }
    public void Beslen()
    {
        Console.WriteLine("Kedi süt içti.");
    }
}
public class Kopek : IHayvan, IBeslen
{
    public void SesCikar()
    {
        Console.WriteLine("Köpek: hav hav");
    }
    public void Beslen()
    {
        Console.WriteLine("Köpek et yedi.");
    }
}
class Program
{
    static void Main(string[] args)
    {
        Kedi kedi = new Kedi();
        kedi.SesCikar();
        kedi.Beslen();
        Kopek kopek = new Kopek();
        kopek.SesCikar();
        kopek.Beslen();
        Console.WriteLine("==================");
        IHayvan hayvan1 = kedi;
        IHayvan hayvan2 = kopek;
        hayvan1.SesCikar();
        hayvan2.SesCikar();
        Console.WriteLine("==================");
        IBeslen beslen1 = kedi;
        IBeslen beslen2 = kopek;
        beslen1.Beslen();
        beslen2.Beslen();
      }
}
```

```
// Ekran çıktısı:

Kedi: miyav
Kedi süt içti.
Köpek: hav hav
Köpek et yedi.
==================
Kedi: miyav
Köpek: hav hav
==================
Kedi süt içti.
Köpek et yedi.
```

Hem Kedi hem de Kopek sınıfları, IHayvan ve IBeslen arayüzlerinden türetilmiştir. Dolayısıyla her iki sınıf da arayüzlerin içindeki metotları uygulamak zorundadır. Arayüzden türetilen sınıflar için soyut sınıflarda olduğu gibi "Her ... bir ... dır." ifadesi yine kullanılır ("Her kedi bir hayvandır.", "Her köpek bir hayvandır."). Tablo 3.2'de arayüzler ve soyut sınıflar arasındaki farklar listelenmiştir.

Tablo 3.2: Arayüzler ve Soyut Sınıflar Arasındaki Farklar

Arayüzler	Soyut Sınıflar
Bir sınıf birden fazla arayüzden türetilebilir.	Bir sınıf sadece tek bir soyut sınıftan türetilebilir.
Sadece boş (gövdesi olmayan) metotlar tanımlanabilir.	Hem normal metot hem de boş metotlar tanımlanabilir.
Çoklu kalıtım özelliği sağlar.	Çoklu kalıtım özelliği sağlamaz.
Tüm ögeler public olarak kabul edilir.	Ögeler public olmak zorunda değildir.
Yapıcı metot içeremez.	Yapıcı metot içerebilir.
Statik ögeler barındıramaz.	Statik ögeler barındırabilir.

Sıra Sizde

Kitap, dergi ve ansiklopediler için bir arayüz aracılığıyla Oku() metotlarını ihtiva etmek zorunda olan sınıfları tanımlayınız.

3.13. ÇOK BİÇİMLİLİK (POLYMORPHISM)

Çok biçimlilik, NTP'deki bir diğer önemli prensiptir. Çok biçimlilik, aynı temel sınıftan veya arayüzden türetilmiş alt sınıflardaki metotların farklı şekillerde davranabilmesidir. Türetilen alt sınıflarda şu durumlardan birinin sağlanması gerekir:

- Üst sınıftaki sanal ögeler geçersiz kılınır (virtual / override).
- Soyut sınıflarda soyut tanımlanan ögeler geçersiz kılınır (abstract / override).
- Arayüzlerdeki ögeler uygulanır (implementation).

Her üç yolla da çok biçimlilik sağlanabilir.

Sınıflar için üç yolla da çok biçimliliğin sağlandığı aşağıda verilmiştir.

virtual / override

```
class Sekil
{
    public virtual void Ciz()
    {
        Console.WriteLine("Şekil çizildi.");
    }
}
class Kare : Sekil
{
    public override void Ciz()
    {
        Console.WriteLine("Kare çizildi.");
    }
}
class Daire : Sekil
{
    public override void Ciz()
    {
      Console.WriteLine("Daire çizildi.");
    }
}
class Ucgen : Sekil
{
    public override void Ciz()
    {
        Console.WriteLine("Üçgen çizildi.");
    }
}
```

abstract / override

```
abstract class Sekil
{
    public abstract void Ciz();
}
class Kare : Sekil
{
    public override void Ciz()
    {
        Console.WriteLine("Kare çizildi.");
    }
}
class Daire : Sekil
{
    public override void Ciz()
    {
        Console.WriteLine("Daire çizildi.");
    }
}
class Ucgen : Sekil
{
    public override void Ciz()
    {
        Console.WriteLine("Üçgen çizildi.");
    }
}
```

interface

```
interface Sekil
{
    void Ciz();
}
class Kare : Sekil
{
    public void Ciz()
    {
        Console.WriteLine("Kare çizildi.");
    }
}
class Daire : Sekil
{
    public void Ciz()
    {
        Console.WriteLine("Daire çizildi.");
    }
}
class Ucgen : Sekil
{
    public void Ciz()
    {
        Console.WriteLine("Üçgen çizildi.");
    }
}
```

Aşağıdaki programda çok biçimlilik uygulanmıştır.

```
class Program
{
    static void Main(string[] args)
    {
        List<Sekil> sekiller = new List<Sekil>
        {
            new Daire(),
            new Kare(),
            new Ucgen()
        };
        foreach (var sekil in sekiller)
        {
            sekil.Ciz();
        }
    }
}
// Ekran çıktısı:
Daire çizildi.
Kare çizildi.
Üçgen çizildi.
```

Burada önemli olan husus, "sekil.Ciz();" ifadesi ile oluşturulan farklı nesnelerin aynı isimli metodunun çağrılarak ilgili nesneye ait işlemlerin gerçekleştirilmesidir.

Sıra Sizde

Kitap, dergi ve ansiklopedileri tutan bir koleksiyon nesnesi oluşturunuz ve bu sınıfların içindeki Oku metodunu döngü kullanarak çağırınız.

3.14. STATİK SINIFLAR (STATIC CLASSES)

Statik (static) sınıflar temel olarak statik olmayan sınıflarla aynıdır ancak statik sınıflardan new anahtar kelimesi ile nesne türetilemez. Nesne türetilemediği için bu sınıfların ögelerine nesne adı üzerinden değil, doğrudan sınıf adı üzerinden erişilir. Sınıfın kendisi ya da içindeki bazı ögeler statik olarak tanımlanabilir. Sınıfın kendisi statik olarak tanımlanırsa sınıf içindeki tüm ögelerin statik olması zorunludur.

Statik ögeler genellikle nesnelerin durumuna göre değişmeyen verileri temsil etmede veya hesaplamaları yapmada kullanılır. Buna en güzel örnek, .NET Sınıf Kütüphanesi'ndeki Math sınıfıdır.

```
Math m1 = new Math(); // Hata
Math m2 = new Math(); // Hata
Console.WriteLine(m1.Sqrt(9));
Console.WriteLine(m2.Sqrt(9));
```

Bir sayının karekökü kod yazılarak hesaplanmak istendiğinde **m1** ve **m2** nesneleri üzerinden hesaplamada farklılık olmayacağı için **Math** sınıfından bir nesne türetilmesine gerek yoktur. Dolayısıyla **Math** sınıfı **static** olarak tanımlanmıştır ve bu sınıftan nesne türetilemez. Bu yüzden kod parçası hata verecektir.

```
double sayi = Math.Sqrt(9);
Console.WriteLine(sayi);
```

Karekök hesaplamak için kullanılan **Sqrt()** metoduna nesne oluşturmadan sınıf adı üzerinden erişilir.

Statik bir sınıf, herhangi bir ögesi ilk kullanıldığı anda hafızaya yüklenir ve programın çalışması sonlana-na kadar hafızada durur. Bu yüzden statik sınıfın dikkatli kullanılması önerilir.

Bir sınıf **static** olarak tanımlanmadan içindeki herhangi bir öge **static** olarak tanımlanabilir. Ayrıca statik bir yapıcı oluşturmak da mümkündür. Aşağıdaki örnekte hem normal bir sınıf yapıcısı hem de statik yapıcı bir arada kullanılmıştır.

```
public class Ogrenci
{
    public int Numara { get; set; }
    public string AdSoyad { get; set; }
}
public class OgrenciIslem
{
    public List<Ogrenci> ogrenciler;
    public static int OgrenciSayisi { get; set; }
    static OgrenciIslem()
    {
        OgrenciSayisi = 0;
        Console.WriteLine("Statik yapıcı çalıştı.");
    }
    public OgrenciIslem()
    {
        ogrenciler = new List<Ogrenci>();
        Console.WriteLine("Yapıcı çalıştı.");
}
    public void OgrenciEkle(Ogrenci ogr)
    {
        ogrenciler.Add(ogr);
        OgrenciSayisi++;
        Console.WriteLine("Öğrenci eklendi.");
 }
```

```
  public void OgrenciSil(int numara)
    {
        var ogr = ogrenciler.FirstOrDefault(x => x.Numara == numara);
        if (ogr != null)
        {
            ogrenciler.Remove(ogr);
            OgrenciSayisi--;
            Console.WriteLine("Öğrenci silindi.");
        }
    }
}
```

Kod blokunun devamı sonraki sayfada

```
internal class Program
{
    private static void Main(string[] args)
    {
        OgrenciIslem oi = new OgrenciIslem();
        Console.WriteLine("1) ==========================");
        oi.OgrenciEkle(new Ogrenci
        {
            Numara = 100,
            AdSoyad = "Nihal Öz"
        });
        Console.WriteLine("Öğrenci sayısı: " + OgrenciIslem.OgrenciSayisi);
        Console.WriteLine("2) ==========================");
        oi.OgrenciEkle(new Ogrenci
        {
            Numara = 200,
            AdSoyad = "İbrahim Yurt"
        });
        Console.WriteLine("Öğrenci sayısı: " + OgrenciIslem.OgrenciSayisi);
        Console.WriteLine("3) ==========================");
        oi.OgrenciSil(100);
        Console.WriteLine("Öğrenci sayısı: " + OgrenciIslem.OgrenciSayisi);
    }
}
// Ekran çıktısı:
Statik yapıcı çalıştı.
Yapıcı çalıştı.
1) ==========================
Öğrenci eklendi.
Öğrenci sayısı: 1
2) ==========================
Öğrenci eklendi.
Öğrenci sayısı: 2
3) ==========================
Öğrenci silindi.
Öğrenci sayısı: 1
```

Sıra Sizde

Bir kütüphane sınıfını aşağıdaki istekleri karşılayacak şekilde oluşturunuz.

1. Eklenen ögelerin adedini tutan gizli bir özelliği olsun.

2. Kitap, dergi veya ansiklopedi ekleme metodu olsun.

3. Eklenen öge adedini ekrana yazdıran bir metodu olsun.

3.15. İSİMSİZ SINIFLAR (ANONYMOUS CLASSES)

İsimsiz sınıflar yalnızca salt okunur özellikleri içeren ve adı olmayan sınıflardır. Alan, metot gibi diğer ögeleri barındıramaz. Sınıf özelliklerinin veri tipleri, aldığı değere göre otomatik olarak belirlenir.

Bir isimsiz sınıf, **var** anahtar kelimesi ile tanımlanır ve **new** anahtar kelimesi ile oluşturulur.

```
var ogrenci = new
{
    Numara = 35,
    Ad = "Yasin",
    Ortalama = 80.5
};
Console.WriteLine("Öğrencinin adı: " + ogrenci.Ad);
//ogrenci.Ortalama = 90.2; HATA!!!
```

Ayrıca bir isimsiz sınıf içinde bir başka isimsiz sınıf oluşturulabilir.

```
var ogrenci = new
{
    Numara = 35,
    Ad = "Yasin",
    Ortalama = 80.5,
    Adres = new
    {
        Il = "Malatya",
        Ilce = "Yeşilyurt"
    }
};
Console.WriteLine("Öğrencinin yaşadığı il: " + ogrenci.Adres.Il);
//ogrenci.Ortalama = 90.2; HATA!!!
```

İstenirse isimsiz sınıf dizisi de oluşturulabilir.

```
var ogrenciler = new[]
        {
        new { Numara = 100, Ad = "Yasin", Ortalama = 80},
        new { Numara = 200, Ad = "İsmail", Ortalama = 75},
        new { Numara = 300, Ad = "Ömer", Ortalama = 60}
    };

Console.WriteLine("2. öğrencinin adı: " + ogrenciler[1].Ad);
Console.WriteLine("======================");
foreach (var ogrenci in ogrenciler)
{
    Console.WriteLine("Adı: {0}, Ortalaması: {1}", ogrenci.Ad, ogrenci.Ortalama);
}
```

```
// Ekran çıktısı:
2.öğrencinin adı: İsmail
==================
Adı: Yasin, Ortalaması: 80
Adı: İsmail, Ortalaması: 75
Adı: Ömer, Ortalaması: 60
```

Sıra Sizde

Bir isimsiz sınıfı parametre olarak alan metodu nasıl tanımlayabileceğinizi araştırınız ve edindiğiniz bilgileri sınıf arkadaşlarınızla paylaşınız.

3.16. MÜHÜRLÜ SINIFLAR (SEALED CLASSES)

Bir sınıftan bir başka sınıf türetilmek istenmediğinde bu sınıfı **sealed** anahtar kelimesiyle mühürlü tanımlamak gerekir.

Yandaki kod parçası hata verecektir (Görsel 3.14).

```
public sealed class UstSinif
{
    // ..
}
public class AltSinif : UstSinif // HATA !!!
{
    // ...
}
```

Görsel 3.14: Derleyici hata mesajı

3.17. PARÇALI SINIFLAR (PARTIAL CLASSES)

Büyük projelerde oluşturulan sınıfları birden fazla dosyaya yaymak mümkündür. Parçalı sınıflar; büyük sınıfları parçalamak, okunmasını kolaylaştırmak, mantıksal olarak katmanlara ayırmak (veri tabanı işlemlerinin ayrı bir dosyada olması gibi), sınıf ögelerini ayrıştırmak (özellikler bir dosyada, metotlar başka bir dosyada vb.), aynı sınıf üzerinde birden fazla programcının çalışması gibi durumlar için kullanılabilir.

Parçalı sınıflar oluşturmak için **partial** anahtar kelimesi kullanılır. Parçalı sınıfların isimleri aynı olmalıdır.

```
// ParcaliSinif1.cs
public partial class ParcaliSinif
{
    public int Ozellik1 { get; set; }
    // ...
}
```

Kod blokunun devamı sonraki sayfada

```
// ParcaliSinif2.cs
public partial class ParcaliSinif
{
    public void Metot1() { }
    //...
}

// Program.cs
internal class Program
{
    private static void Main(string[] args)
    {
        ParcaliSinif ps = new ParcaliSinif();
        Console.WriteLine(ps.Ozellik1);
        ps.Metot1();
    }
}
```

Program derlendiğinde tüm parçalar birleştirilir ve tek bir sınıf tanımlanmış gibi çalıştırılır.

3.18. ENUMS (NUMARALANDIRMALAR)

Bir **enum** (enumerations kelimesinin kısaltması) sadece **int** tipindeki sabitlerden oluşan özel bir sınıftır. Bu değerler sadece okunabilirdir ve değiştirilemez. Enum genellikle programların okunmasını kolaylaştırmak için kullanılır.

Bir numaralandırma oluşturabilmek için enum anahtar kelimesi ve değerleri birbirinden ayırmak için virgül (,) karakteri kullanılır.

```
enum Seviyeler
{
    Çok_Düşük, // 0
    Düşük,     // 1
    Orta,      // 2
    Yüksek,    // 3
    Çok_Yüksek // 4
}
internal class Program
{
    private static void Main(string[] args)
    {
        Console.WriteLine(Seviyeler.Düşük);
        Console.WriteLine((int)Seviyeler.Düşük);
    }
}
// Ekran çıktısı:
Düşük
1
```

enum içindeki değerler 0'dan (sıfır) başlayarak birer birer artar. İstenirse farklı tam sayı değerleri de verilebilir.

```
enum Kategoriler
{
    Bilgisayar = 3,
    Mobilya    = 10,
    Kırtasiye  = 7,
    Hırdavat,   // 8
    Otomobil    // 9
}
```

Örnekte bilgisayar, mobilya ve kırtasiye kategorilerine istenilen değerler atanmıştır. Değer ataması yapılmayan hırdavat ve otomobil kategorilerine ise 7'den sonra gelen 8 ve 9 değerleri otomatik olarak atanmıştır.

Program içinde örnek kullanım aşağıda verilmiştir.

```
// enum tanımlama
Kategoriler kat = Kategoriler.Kırtasiye;
// değer adının kullanımı
Console.WriteLine(kat);
// değerin sayısal değerinin kullanımı
Console.WriteLine((int)kat);
// enuma sayısal değer atama
kat = (Kategoriler)8;
// if ile kullanımı
if (kat == Kategoriler.Kırtasiye)
    Console.WriteLine("Hırdavat kategorisi");
// switch ile kullanımı
switch (kat)
{
    case Kategoriler.Bilgisayar: // ..
        break;
    case Kategoriler.Mobilya:    // ..
        break;
    case Kategoriler.Kırtasiye:  // ..
        break;
    case Kategoriler.Hırdavat:   // ..
        break;
    case Kategoriler.Otomobil:   // ..
        break;
    default:
        break;
}
```

ÖLÇME VE DEĞERLENDİRME

A) Aşağıdaki işlemleri gerçekleştiriniz.

1. "Ses seviyesi", "ekran boyutu" ve "görüntü teknolojisi" alanlarına sahip bir "Televizyon" sınıfı yazınız.

2. "RAM bellek kapasitesi", "CPU", "HD kapasitesi" alanlarına sahip bir "Bilgisayar" sınıfı yazınız.

3. "Televizyon" sınıfını özellikler kullanarak tekrar yazınız.

4. "Bilgisayar" sınıfını özellikler kullanarak tekrar yazınız.

5. "Televizyon" sınıfına "Güç aç / kapat", "Kanal değiştir" ,"Ses seviyesi oku" metotlarını ekleyiniz (Gerekli alanları sınıfa ekleyiniz.).

6. "Televizyon" sınıfının "Kanal değiştir" metodunu aşağıdaki şekilde güncelleyiniz (Gerekli alanları sınıfa ekleyiniz.). Eklediğiniz metotları program içinde kullanınız.

- KanalNoArtir() => Kanal numarasını bir artırmalı.
- KanalNoArtir(int) => Kanal numarasını parametrede verilen değer kadar artırmalı.
- KanalNoAzalt() => Kanal numarasını bir azaltmalı.
- KanalNoAzalt(int) => Kanal numarasını parametrede verilen değer kadar azaltmalı.

7. "Televizyon" sınıfında kullandığınız alanların, özelliklerin ve metotların erişim türlerini açıklayınız.

8. "Televizyon" sınıfından "İşletim sistemi" özelliğine sahip bir "Akıllı televizyon" sınıfını aşağıdaki hususlara dikkat ederek türetiniz.

- "Televizyon" sınıfındaki "Güç aç / kapat" metodu bu sınıf içinde tekrar yazılmalıdır.
- "Televizyon" sınıfındaki "ses seviyesi" bilgisi sadece bu iki sınıf içinde kullanılabilir olmalıdır.

B) Aşağıdaki açık uçlu soruları cevaplandırınız.

9. "Televizyon" sınıfından türetilen tüm nesneler için "Marka" bilgisinin aynı olması istenirse bu sınıf üzerinde nasıl bir değişiklik yapılır?

10. "Televizyon" sınıfından başka bir sınıf türetilemesin istenirse bu sınıf üzerinde nasıl bir değişiklik yapılır?

C) Aşağıdaki cümlelerde boş bırakılan yerlere doğru sözcükleri yazınız.

11. Sınıf içindeki bir değişkeni dış dünyaya kapatıp sadece sınıf içinde kullanılabilir kılmak için bu özellik ..... şeklinde tanımlanmalıdır.

12. Sınıf içindeki bir değişkeni dış dünyaya kapatıp sadece sınıf içinde ve bu sınıftan türetilen alt sınıflarda kullanılabilir kılmak için bu özellik ..... şeklinde tanımlanmalıdır.

13. Sınıf içindeki bir değişkeni her yerden erişilebilir kılmak için bu özellik ..... şeklinde tanımlanmalıdır.

14. Aşağıdaki kod çalıştırıldığı zaman ekranda üreteceği çıktı nedir? Boş kutucuğa yazınız.

```
using System;
class program
{
    static void Main(string[] args)
    {
        int num = 2;
        Fonk1(ref num);
        Console.WriteLine(num);
    }
    static void Fonk1(ref int num)
    {
        num = num * num * num;
    }
}
```

15. Sınıf içinde nesne oluşturulurken ve nesne hafızadan atılırken otomatik olarak çalıştırılan metotlar nasıl adlandırılır? Özellikleri nelerdir? Açıklayınız.

16. Aşağıdaki kod çalıştırıldığı zaman ekranda üreteceği çıktı nedir? Boş kutucuğa yazınız.

```
static void Main(string[] args)
{
    int sayi = 5;
    int kare = 0, kup = 0;
    Hesapla(sayi, kare, ref kup);
    Console.WriteLine(kare + " & " + kup);
    Console.ReadLine();
}
static void Hesapla(int sayi, int kare, ref int kup)
{
    kare = sayi * sayi ;
    kup = kare * sayi;
}
```

17. Statik sınıflar hangi durumlarda kullanılır?

18. Statik yapıcı metotlar ne zaman çalıştırılır?

19. Soyut sınıf ile arayüzler arasındaki benzerlikler ve farklılıklar nelerdir?

20. "Güç aç" ve "Güç kapat" metotlarını tanımlayan bir arayüz yazarak "Televizyon" ve "Bilgisayar" sınıflarına bu metotları uygulayınız.

4. ÖĞRENME BİRİMİ

DİZİLER (ARRAYS) VE KOLEKSİYONLAR (COLLECTIONS)

KONULAR

4.1. SINIFLAR VE NESNELER
4.2. DİZİLER
4.3. KOLEKSİYONLAR

ANAHTAR KELİMELER

Dizi, index, veri tipi, for döngüsü, foreach döngüsü, koleksiyon

NELER ÖĞRENECEKSİNİZ?

- Dizi kavramı
- Dizi tanımlama
- Dizilere değer verme ve dizilerden değer alma işlemleri
- Çok boyutlu dizi kavramı
- Çok boyutlu dizi tanımlama
- Çok boyutlu dizilere değer verme ve dizilerden değer alma işlemleri
- İhtiyaca uygun olarak dizileri kullanma
- Koleksiyon kavramı
- Koleksiyon tanımlama
- Koleksiyonlara değer verme ve koleksiyonlardan değer alma işlemleri
- İhtiyaca uygun koleksiyon kullanımı

HAZIRLIK ÇALIŞMALARI

1. Projenizde girilen 10 adet sayının ortalamasını değişkenler kullanarak nasıl alırsınız? Sınıf arkadaşlarınızla paylaşınız.
2. C# programlama dilinde koleksiyon kavramını ve avantajlarını araştırınız.

4.1. DİZİLER

Dizi, aynı tipte birden çok değeri bellek üzerinde tutabilecek yapıdır. Programlama yaparken dizileri kullanmak; dizilerin verdiği avantajlardan yararlanarak değerler üzerinde seçme, silme, değiştirme, sıralama vb. işlemlerin kolayca gerçekleştirilmesini sağlar.

4.1.1. Tek Boyutlu Diziler

Bir boyutlu veya tek boyutlu diziler, verileri saklamak için bir satırdan oluşan dizilerdir. Bu diziler, art arda sıralanmış bellek alanları gibi düşünülebilir. Aynı tipten değerler olmak şartıyla belirlenen adet kadar veri, dizi içinde sıralanmış bellek alanlarında saklanır. Burada dikkat edilmesi gereken noktaların başında diziyi oluştururken içinde kaç adet veri olacağı ve dizilerde hangi tip verilerin saklanacağı (int, string, char, double vb.) gelir. Belirtilen tipin sınırı dışında veya belirtilen adet sayısından daha fazla veri saklamaya çalışıldığında derleyici hata verir.

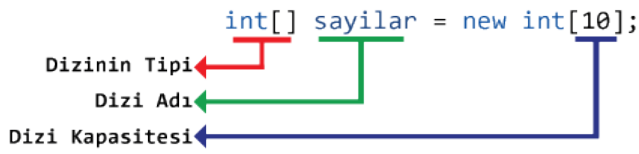
4.1.2. Bir Boyutlu Dizilerin Oluşturulması

Dizi oluştururken temelde üç noktaya dikkat edilir.

Dizinin Tipi: Dizide hangi tip verilerin saklanacağı (int, string, char, byte, double vb.) belirtilir.

Dizi Adı: Dizide saklanacak verilerle anlamlandırılan değişken isimlendirme kurallarına uygun hangi isimlerin diziye verileceği belirtilir (Anlamlı isimler vermek, yazılan kodların okunabilirliğini artıracağı için her zaman tavsiye edilen bir yöntemdir. Örneğin okul numaraları saklanacak bir dizi için diziOkulNo kullanılabilir.).

Dizilerin Kapasitesi: Dizi içinde kaç adet veri saklanacağı belirtilir. Görsel 4.1'de sayilar isminde, integer tipinde 10 adet veri saklama kapasitesine sahip bir dizi tanımlaması yapılmıştır. Derleyicinin bir diziyi tanıması için başlangıçta veri tipi belirtildikten sonra içi boş köşeli parantezler kullanılmalıdır. İçi boş köşeli parantezler, bu ifadenin bir boyutlu dizi olduğu anlamına gelir. Tanımlamadaki ikinci köşeli parantez ise dizide saklanacak değer sayısını belirtir. Aşağıda farklı veri tiplerine sahip dizi tanımlama örnekleri verilmiştir.



Görsel 4.1: Bir boyutlu dizi tanımlaması

```
string[] isimler = new string[5];  // String tipinde 5 elemanlı dizidir.
byte[] siralar = new byte[6];      // Byte tipinde 6 elemanlı dizidir.
bool[] durumlar = new bool[4];     // Boolean tipinde 4 elemanlı dizidir.
float[] uzunluklar = new float[8]; // Float tipinde 8 elemanlı dizidir.
```

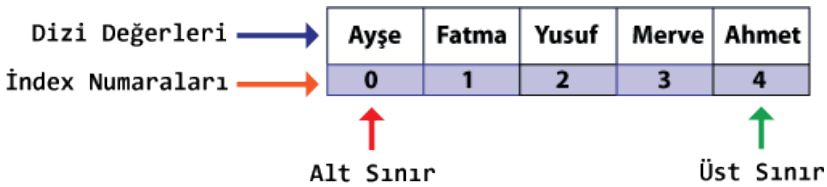
Bir dizi tanımlaması yapıldığında derleyici, dizinin her elemanına temel veri tipleri için varsayılan değerleri ilk değer olarak verir. İlk değerler, dizi içine veri eklenmeden verilir. Bunlar; string tipi için null, sayısal tipler için 0, bool tipi için ise false değerleridir. Verilen bu ilk değerler, dizilere değer aktarımı yapıldıkça yeni değerlerle değiştirilir.

4.1.3. Bir Boyutlu Dizilere Değer Aktarma

Dizilere değer aktarmanın farklı yöntemleri vardır. Dizilere ilk olarak tanımlamasının yapıldığı satırda değer verilebilir.

```
string[] kisiler = new string[5] {"Ayşe","Fatma","Yusuf","Merve","Ahmet"};
```

Tanımlanan ve aynı satırda değer aktarımı yapılan dizide kodlar derlendiğinde bellekte **5** elemanlı bir dizi oluşturulur. Oluşturulan bu diziye küme parantezi **{ }** içindeki değerler sırasıyla verilir.



Görsel 4.2: Diziye değer aktarma

Günlük hayatta sıralama işlemlerine hep 1'den başlanır fakat programlama dillerinin çoğunda sıralama 0'dan başlar. Görsel 4.2'de eklenen ilk elemanın sıra numarası 0'dır. Dizilerin her değerinin bir sıra numarası vardır. Sıra numaraları index, indis veya indeks olarak adlandırılır.

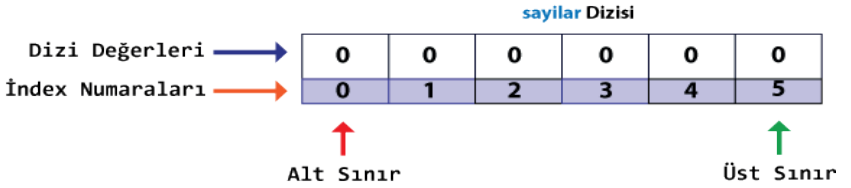
Tanımlandıkları satırda dizilere değer aktarma işlemi farklı şekillerde de yapılabilir. Aşağıdaki örnekte diziye değer aktarım işlemi, dizinin eleman sayısı belirtilmeden veya new sözcüğü kullanılmadan gerçekleştirilmiştir. Bu durumda derleyici hata vermez ve dizinin eleman sayısı derleyici tarafından belirlenir.

```
string[] kisiler = new string[] {"Ayşe","Fatma","Yusuf","Merve","Mehmet"};
string[] kisiler = {"Ayşe","Fatma","Yusuf","Merve","Mehmet"};
```

Dizilere değer aktarımının bir diğer yöntemi, dizinin index numaralarının kullanılarak yapılmasıdır.

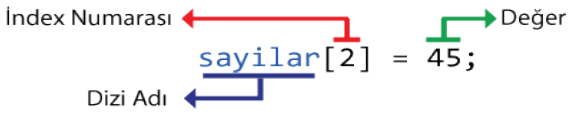
```
int[] sayilar = new int[6];
```

Kodda **integer** veri tipine sahip, **6** elemanlı, **sayilar** adında bir dizi tanımlandı. Derleyici, bu kod satırına geldiğinde bellekte eleman sayısı kadar yer ayırır ve bu yerlere ilk değer olarak **0** (sıfır) sayısını aktarır.



Görsel 4.3: Dizinin ilk değerlerinin verilmesi

Bellek üzerinde dizi oluşturulduktan sonra index numaraları kullanılarak değer aktarımı gerçekleştirilebilir. Görsel 4.4'te **sayilar** ismindeki dizinin **2** numaralı index elemanına (dizinin üçüncü elemanına) 45 değerinin aktarımı yapılmıştır.



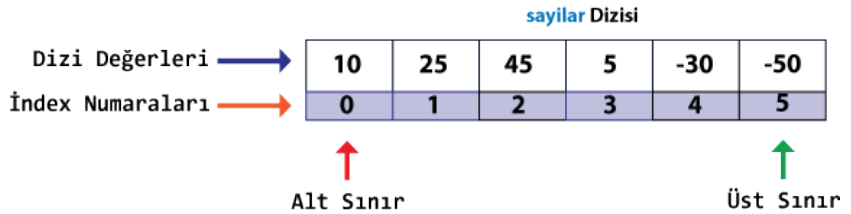
Görsel 4.4: Diziye index numarası ile değer aktarılması

Sıra Sizde

Aşağıdaki kodlamaları yaparak dizi elemanlarına değer aktarma işlemini gerçekleştiriniz.

```
int[] sayilar = new int[6];
sayilar[0] = 10;  //sayilar dizisinin 0 index numaralı elemanı 10 oldu.
sayilar[1] = 25;  //sayilar dizisinin 1 index numaralı elemanı 25 oldu.
sayilar[2] = 45;  //sayilar dizisinin 2 index numaralı elemanı 45 oldu.
sayilar[3] = 5;   //sayilar dizisinin 3 index numaralı elemanı 5 oldu.
sayilar[4] = -30; //sayilar dizisinin 4 index numaralı elemanı -30 oldu.
sayilar[5] = -50; //sayilar dizisinin 5 index numaralı elemanı -50 oldu.
```

Görsel 4.5'te değer aktarma işlemi bittikten sonra derleme işleminde bellek üzerindeki dizinin son hâli verilmiştir.



Görsel 4.5: Değer atandıktan sonra dizinin bellek üzerindeki durumu

Değer aktarım işleminde **index** numaralarına göre dizinin index numarası **0**'dan başlayarak dizideki eleman sayısının bir eksiğine kadar istenilen alana değer aktarılabilir. Tanımlanan dizide **0**'dan küçük ve eleman sayısının bir eksiğinden büyük bir index numarası ile diziye değer aktarmaya veya dizi elemanına erişmeye çalışıldığında derleyici tarafından hata mesajı gönderilir. Hata mesajı, girilen index numarasının dizinin sınırları dışında olduğunu bildirir.

```
isayilar[6] = -30;
```

Kod yazıldığında derleyici, Görsel 4.6'daki hata mesajını vererek kullanıcıyı uyarır.

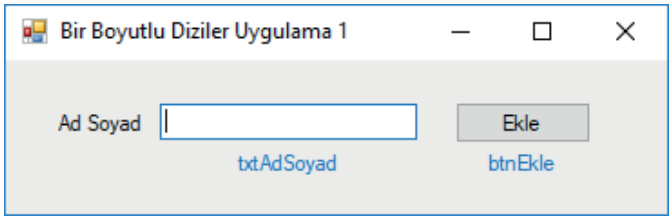
Görsel 4.6: Dizi sınır aşımı hata mesajı

1. Uygulama

Bir Boyutlu Diziler

Dizilere değer aktarım işlemi, kodlama sırasında değil de uygulamanın çalışması esnasında olabilir. Aşağıdaki uygulamada çalışma esnasında dizilere değer aktarım işlemi yapılmıştır.

1. Adım: Görsel 4.7'deki form tasarımını yaptıktan sonra form içindeki kontrollere name değerlerini (mavi yazı ile belirtilen) veriniz.



Görsel 4.7: Dizi uygulamaları form tasarımı

2. Adım: Ekle butonu Click olayında butona her tıkladığınızda TextBox içine girilen değerleri isimler adındaki 5 elemanlı bir diziye aktaracak kodlamaları yapınız.

```
string[] isimler = new string[5]; //Global Dizi
int index = 0; //Global Değişken
private void btnEkle_Click(object sender, EventArgs e)
{
    isimler[index] = txtAdSoyad.Text;
    index++;
}
```

Sıra Sizde

1. Kodlarda isimler dizisi neden btnEkle_Click içinde değil de global olarak tanımlanmıştır? Arkadaşlarınızla paylaşınız.
2. İkinci adımda verilen kodda tanımlanan index ismindeki değişken neden kullanılmıştır? Açıklayınız.
3. Uygulamada 6. kişi eklenmeye çalışıldığında nasıl bir hatayla karşılaşılır? Karşılaşılan bu hatanın çözümü için neler yapılabileceğini sınıf arkadaşlarınızla paylaşınız.

4.1.4. Bir Boyutlu Dizi Elemanlarına Erişim

Dizinin elemanlarına erişim, dizilere değer aktarmada olduğu gibi index numaraları kullanılarak sağlanır. Erişim sağlanan dizi elemanı; değişkenlere aktarma, hesaplamalar yapma, ekrana yazdırma, nesneye aktarma gibi işlemlerde kullanılır.

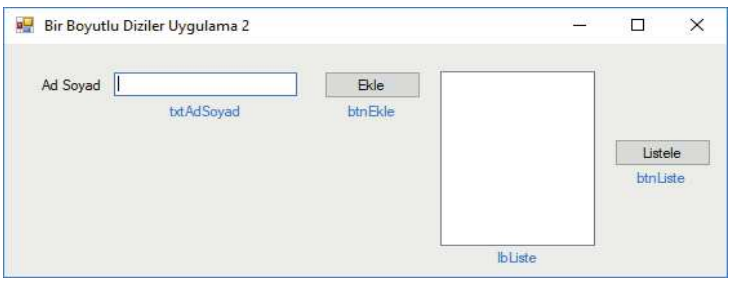
```
int[] dizi = new int[5] { 46, 41, 34, -10, 55 };
// 5 elemanlı bir dizi tanımlaması yapıldı. Aynı satırda değerler verildi.

int sayi1 = dizi[0];
// Dizinin 0 index numaralı değeri sayi1 ismindeki int tipindeki değişkene aktarıldı.

int toplam = dizi[0] + dizi[1] + dizi[2];
// Dizinin ilk üç elemanıyla toplama işlemi yapıldı.

Console.WriteLine(dizi[1]);
// Dizinin 1 index numaralı değeri (41 değeri) Console ekranına yazdırıldı.

dizi[2] = dizi[3];
// Dizinin 3 index numaralı değeri 2 index numaralı alana aktarıldı.

label1.Text = dizi[4].ToString();
// Dizinin 4 index numaralı değeri label1 nesnesinin text özelliğine aktarıldı.
```

2. Uygulama

Bir Boyutlu Diziler

Bu işlem, birinci uygulamadaki 5 elemanlı isimler dizisinde yer alan değerleri, Listele butonuna tıklayarak ListBox nesnesinin içinde göstermektir.

1. Adım: Görsel 4.8'de verilen formun tasarımını yapınız. Form kontrol nesnelerine name değerlerini veriniz.



Görsel 4.8: Dizi uygulamaları form tasarımı

2. Adım: Listele butonu Click olayına dizi içindeki her bir elemanı for döngüsü yardımıyla ListBox içinde göstermek için Items.Add() metodu kullanarak listeleme işlemi yapan kodlamaları yazınız.

```
private void btnListele_Click(object sender, EventArgs e)
{
  for (int i = 0; i < isimler.Length; i++)
  {
     lbListe.Items.Add(isimler[i]);
  }
}
```

Sıra Sizde

1. Kodlarda dizi isminden sonra **Length** ifadesi hangi amaç için kullanılmıştır?
2. Uygulamada 5 elemanlı dizinin tamamına değer aktarmadan listele butonuna tıklandığında nasıl bir hata ile karşılaşılır? Karşılaşılan bu hatanın çözümü için ne yapılabilir?
3. Dizi değerlerini listeleme işlemi, **for** döngüsü yerine **while** döngüsü kullanılarak nasıl yapılır?

4.1.5. Dizilerde Foreach Döngüsü Kullanımı

Birçok programlama dili, diziler üzerinde işlem yapılmasını kolaylaştıran bir döngü sunar. Bu döngü, foreach döngüsüdür. Dizilerde kullanılan foreach, dizi elemanlarını ilk elemandan başlayıp, dizinin son elemanına kadar her elemanı tek tek dolaşarak belirlenen bir değişkene aktarır. Örneğin 10 elemanlı bir dizide foreach döngüsü kullanıldığında döngü 10 defa tekrarlama işlemi yapar. Döngü her seferinde dizi içindeki değeri alarak aynı tipte olan bir değişkene aktarır. Döngünün yapısı aşağıda verilmiştir.

```
foreach (Tip Değişken in Dizi)
{
  // Döngü içindeki işlemler
}
```

Foreach döngüsünün yapısındaki ögeler aşağıda sıralanmıştır.

Tip: Dizi içindeki veri tipleri ile aynı olmalıdır (Dizi içindeki değerler string ise Tip de string, double ise Tip de double olmalıdır.). Bazı durumlarda Tip olarak **var** kullanılır. Var tipi, kendisine atanan değer ne ise o değerin tipini alır.

Değişken: Döngü, dizi içindeki değeri her dönme işleminde belirtilen bir değişkene aktarır.

in: Bir anahtar kelimedir, foreach döngülerinde değişken adlarından sonra kullanılır.

Dizi: Üzerinde işlem yapılacak dizinin adıdır.

```
int[] sayilar = { 20, 30, 40, 50 };
foreach (int sayi in sayilar)
{
  Console.WriteLine(sayi);
}
```

```
int[] sayilar = { 20, 30, 40, 50 };
for (int i = 0; i < sayilar.Length; i++)
{
  Console.WriteLine(sayilar[i]);
}
```

Verilen örneklerde iki döngü de aynı görevi yerine getirir. Dizilerde for döngüsüne göre daha az kod yazarak sonuca ulaşıldığı için genellikle foreach döngüsü tercih edilir. Foreach döngüsünde dizi içindeki değerler döngü tamamlanıncaya kadar sırasıyla sayi ismindeki değişkene aktarılır. For döngüsünde i değişkeni, dizinin index numarası olarak dizi elemanlarına erişim için kullanılır.

Sıra Sizde

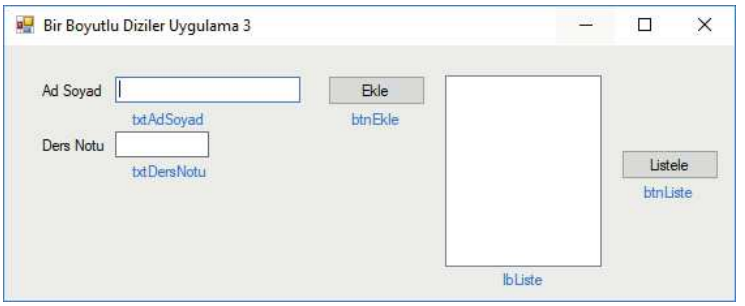
İkinci uygulamadaki dizi elemanlarını listele butonuna tıklama olayında ListBox içine foreach döngüsü kullanarak listeleyiniz.

3. Uygulama

Bir Boyutlu Diziler

Bu işlemde ikinci uygulamadaki 5 elemanlı isimler dizisiyle birlikte integer tipindeki değerleri saklayan 5 elemanlı **notlar** isimli dizi kullanılacaktır. **Listele** butonuna tıklandığında her iki dizideki değerlerin ListBox nesnesinin içinde gösterilmesi sağlanacaktır.

1. Adım: Görsel 4.9'da verilen formun tasarımını yapınız. Form kontrol nesnelerine name değerlerini veriniz.



Görsel 4.9: Dizi uygulamaları form tasarımı-3

2. Adım: Kullanılacak dizileri ve index değişkenini global olarak oluşturunuz.

```
string[] isimler = new string[5]; //Global dizi
  int[] notlar = new int[5];       //Global dizi
  int index = 0;                   //Global değişken
```

3. Adım: Ekle butonunun Click olayına aşağıdaki kodları yazarak dizilere ad, soyad ve ders notu bilgilerini aktarınız.

```
private void btnEkle_Click(object sender, EventArgs e)
{
   if(index<isimler.Length)
   {
      isimler[index] = txtAdSoyad.Text;
      notlar.[index] = int.Parse(txtDersNotu.Text);
      // int.Parse fonksiyonu girilen değeri "int" veri türüne dönüştürür
      index++;
      txtAdSoyad.Text = "";
      txtDersNotu.Text = "";
   }
}
```

4. Adım: Listele butonu Click olayında aşağıdaki kodları yazarak dizideki değerleri ListBox kontrolü içinde gösteriniz.

```
private void btnListele_Click(object sender, EventArgs e)
 {
   for (int i = 0; i < isimler.Length; i++)
   {
     if(isimler[i] != null)
     {
       lbListe.Items.Add(isimler[i]+" > "+notlar[i]);
     }
   }
 }
```

Sıra Sizde

1. Üçüncü adımda karşılaştırma ifadesinin kullanım amacı nedir?

2. Dördüncü adımda karşılaştırma ifadesinin kullanım amacı nedir?

3. Üçüncü adımda for döngüsü yerine foreach döngüsü kullanılabilir mi? Neden?

4. Uygulama

http://kitap.eba.gov.tr/KodSor.php?KOD=21077

Bir Boyutlu Diziler

Bu işlem, üçüncü uygulamanın geliştirilmiş hâlidir. Bir önceki uygulamada isimler ve notlar dizilerine değer aktarımı sağlanmıştı. Bu uygulamada ise girilen notlar içinde en yüksek ve en düşük not ile tüm notların ortalamasını hesaplama işlemi yapılacaktır.

1. Adım: Görsel 4.10'da verilen formun tasarımını yapınız. Form kontrol nesnelerine name değerlerini veriniz.

Görsel 4.10: Dizi uygulamaları form tasarımı-4

2. Adım: En Yüksek butonu Click olayında notlar dizisi içindeki en yüksek notu bulduran kodlamayı yapınız.

```
private void btnEnYuksek_Click(object sender, EventArgs e)
{
   int enyuksek = notlar[0];
   for (int i = 0; i < notlar.Length; i++)
   {
       if (notlar[i] > enyuksek)
       {
         enyuksek = notlar[i];
       }
   }
   txtEnYuksek.Text = enyuksek.ToString();
}
```

3. Adım: En Düşük butonu Click olayında notlar dizisi içindeki en düşük notu bulduran kodlamayı yapınız.

```
private void btnEnDusuk_Click(object sender, EventArgs e)
 {
    int endusuk = notlar[0];
    for (int i = 0; i < notlar.Length; i++)
    {
       if (notlar[i] < endusuk)
          endusuk = notlar[i];
    }
    txtEnDusuk.Text = endusuk.ToString();
}
```

4. Adım: Ortalama butonu Click olayında notlar dizisi içindeki notların ortalamasını bulduran kodlamayı yapınız.

```
 private void btnOrtalama_Click(object sender, EventArgs e)
 {
    int toplam=0;
    double ortalama=0;
    for (int i = 0; i < notlar.Length; i++)
    {
        toplam += notlar[i];
     }
     ortalama = toplam / notlar.Length;
  txtOrtalama.Text = ortalama.ToString();
}
```

Sıra Sizde

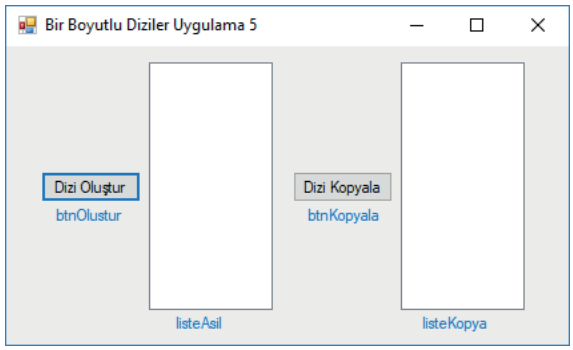
1. İkinci adımın çalışma mantığının nasıl olduğunu açıklayınız.
2. İkinci adımda kullandığınız kodda neden dizinin ilk elemanı bir değişkene aktarıldı?
3. İkinci ve üçüncü adımdaki karşılaştırma ifadelerinin kullanım amacı nedir?
4. Dördüncü adımda toplam ve ortalama değişkenlerine neden başlangıç değeri olarak 0 verildi?
5. Dördüncü adımda ortalama değişkeninin tipi neden double olarak seçildi?

5. Uygulama

Bir Boyutlu Diziler

Bu uygulamada bir dizinin elemanları başka bir diziye kopyalanacaktır.

1. Adım: Görsel 4.11'de verilen formun tasarımını yapınız. Form kontrol nesnelerine name değerlerini veriniz.



Görsel 4.11: Dizi uygulamaları form tasarımı-5

2. Adım: Formda kaynak dizi ve kopyalanacak diziyi 100 elemanlı ve integer tipinde sayıları saklayacak şekilde global olarak oluşturunuz.

```
int[] diziKaynak = new int[100];
int[] diziKopya = new int[100];
```

3. Adım: Dizi Oluştur butonu Click olayında kaynak diziye 0 ile 100 arasında rastgele sayılardan oluşan değerlerin aktarımı yapılacaktır. Bu işlem için Random sınıfından üretilen rastgele isimli nesneye Next metodu ile birlikte 0 ile 100 arasında üretilen sayıları diziye aktaran ve dizi değerlerini listeleyen kodlamayı yapınız.

```
private void btnOlustur_Click(object sender, EventArgs e)
{
   Random rastgele = new Random();
   for (int i = 0; i < diziKaynak.Length; i++)
   {
      diziKaynak[i] = rastgele.Next(0, 101);
}
   for (int i = 0; i < diziKaynak.Length; i++)
   {
      listeAsil.Items.Add(diziKaynak[i]);
   }
}
```

4. Adım: Dizi Kopyala butonu Click olayında ise kaynak dizideki değerleri kopyalanacak diziye aktarım işlemini yaptıktan sonra kopyalanan diziyi listeleyen kodlamaları yapınız.

```
private void btnKopya_Click(object sender, EventArgs e)
{
    for (int i = 0; i < diziKopya.Length; i++)
    {
        diziKopya[i] = diziKaynak[i];
    }
    for (int i = 0; i < diziKopya.Length; i++)
    {
        listeKopya.Items.Add(diziKopya[i]);
    }
}
```

Sıra Sizde

1. Listeleme işlemlerini foreach döngüsü ile gerçekleştiriniz.
2. Dizi kopyalama işleminin daha kısa yolu var mıdır? Oluşturacağınız küçük gruplarla dizi kopyalama işleminin kısa yolunun olup olmadığını tartışınız. Sonuçları sınıfla paylaşınız.

4.1.6. Bir Boyutlu Dizilerde Kullanılan Özellikler ve Metotlar

Dizilerle işlem yaparken işleri kolaylaştıracak bazı metotlar mevcuttur. Bu metotlar, dizilerin daha etkin kullanılabilmesine olanak sağlar.

Length Özelliği: Dizideki eleman sayısını verir. Bu özelliğin kullanımı **diziadi.Length** şeklindedir.

Rank Özelliği: Dizideki boyut sayısını verir. Bu özelliğin kullanımı **diziadi.Rank** şeklindedir.

Max Metodu: Dizideki en büyük sayıyı verir. Bu metodun kullanımı **diziadi.Max()** şeklindedir.

Min Metodu: Dizideki en küçük sayıyı verir. Bu metodun kullanımı **diziadi.Min()** şeklindedir.

Sum Metodu: Dizideki sayıların toplamını verir. Bu metodun kullanımı **diziadi.Sum()** şeklindedir.

Average Metodu: Dizideki sayıların ortalamasını verir. Bu metodun kullanımı **diziadi.Average()** şeklindedir.

First Metodu: Dizideki ilk elemanı verir. Bu metodun kullanımı **diziadi.First()** şeklindedir.

Last Metodu: Dizideki son elemanı verir. Bu metodun kullanımı **diziadi.Last()** şeklindedir.

Sıra Sizde

Bir form tasarımı yapınız. Formun içine bir adet Button ve bir adet ListBox ekleyiniz. Butona tıklandığında oluşturduğunuz sayısal tipteki diziye 0 ile 10 arasında rastgele değerler aktarınız. Değer aktarımı yapılan dizi için hangi metot ve özelliklerin kullanılabileceğini sınıfla paylaşınız. Elde edilen sonuçları ListBox içinde gösteren kodlamayı yapınız.

4.1.7. Çok Boyutlu Diziler

Tek boyutlu diziler, aynı veri tipinden değerler içeren ve tek satırdan oluşan dizilerdir. Çok boyutlu diziler ise tek boyutlu dizilerin genişletilmiş biçimidir. Çok boyutlu diziler genellikle matematiksel hesaplamalar, görüntü işleme ve kayıt işlemlerinde kullanılır. Çok boyutlu diziler içinde en çok iki boyutlu ve üç boyutlu diziler kullanılır.

İki boyutlu diziler, tek boyutlu dizilerin birden çok satırdan oluşmuş hâlidir. Satranç tahtası, iki boyutlu dizilere örnek olarak verilebilir. Satranç tahtasında 8 satır ve 8 sütundan oluşan kareler ile her karenin satır ve sütunlarının kesişmesinden oluşan bir adresi vardır. Bu adresler kullanılarak karenin tahta üzerindeki konumu belirlenir. Tahtanın her karesinde bir veri saklanacağı varsayılırsa adresler kullanılarak karelerin içindeki verilerle işlem yapılabilir.

4.1.8. İki Boyutlu Dizi Tanımlama

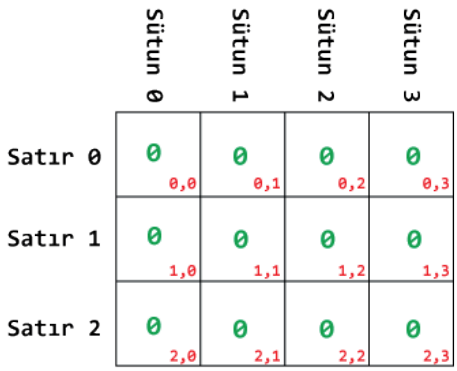
Çok boyutlu dizilerin tanımlanmasında boyut sayısı, veri tipi belirtildikten sonra köşeli parantez içine yazılan virgül adediyle belirlenir. Örneğin hiç virgül yoksa tek boyutlu dizi, bir virgül varsa iki boyutlu dizi, iki virgül varsa üç boyutlu dizi tanımlanacaktır.

```
int[,] dizi2d;          // İki boyutlu dizi
int[, ,] dizi3d;        // Üç boyutlu dizi
int[, , ,] dizi4d ;     // Dört boyutlu dizi
int[, , , ,] dizi5d;    // Beş boyutlu dizi
```

Görsel 4.12'de double tipinde sayılar saklayan **notlar** isminde **3** satır ve **4** sütunlu iki boyutlu bir dizi tanımlaması yapılmıştır. Derleyici, kodu derlediğinde bellek üzerinde 3x4=12 adet sayının saklanacağı bir dizi oluşturur.

Görsel 4.12: İki boyutlu dizi tanımlaması

Derleyici, tanımlanan diziyi derlediğinde bellek üzerindeki bu alanlara varsayılan olarak 0 değerini verir. Görsel 4.13'te sağ alt köşede kırmızı ile belirtilen adresler kullanılarak dizideki verilerle işlem yapılır.Tek boyutlu dizilerde olduğu gibi iki boyutlu dizilerde de hem satır hem sütun sıralaması daima 0'dan başlar.



Görsel 4.13: İki boyutlu dizinin satır ve sütunları

4.1.9. İki Boyutlu Diziye Değer Aktarma

İki boyutlu dizilere iki şekilde değer aktarılır.

1. İki boyutlu diziye tanımlandığı satırda değer aktarımı yapılabilir.

```
double[,] notlar = new double[3,4] {{45,55,60,65},{75,80,85,90},{10,20,30,40}};
```

2. Dizinin index numaraları kullanılarak değer aktarımı yapılabilir.

```
double[,] notlar = new double[3, 4];
  notlar[0, 0] = 45;
  notlar[0, 1] = 55;
  notlar[0, 2] = 60;
  notlar[0, 3] = 65;
  notlar[1, 0] = 75;
  notlar[1, 1] = 80;
  notlar[1, 2] = 85;
  notlar[1, 3] = 90;
  notlar[2, 0] = 10;
  notlar[2, 1] = 20;
  notlar[2, 2] = 30;
  notlar[2, 3] = 40;
```

Diziye değer aktarımı yapıldıktan sonra dizinin değerleri Görsel 4.14'teki gibi olur.

Sıra Sizde

Sınıfınızda üçer kişilik dört takım kuracağınızı düşününüz. İki boyutlu dizi oluşturarak her satırda bir takım bulunacak şekilde arkadaşlarınızın isimlerini bu diziye aktarma işlemini yapınız.

	Sütun 0	Sütun 1	Sütun 2	Sütun 3
Satır 0	45 (0,0)	55 (0,1)	60 (0,2)	65 (0,3)
Satır 1	75 (1,0)	80 (1,1)	85 (1,2)	90 (1,3)
Satır 2	10 (2,0)	20 (2,1)	30 (2,2)	40 (2,3)

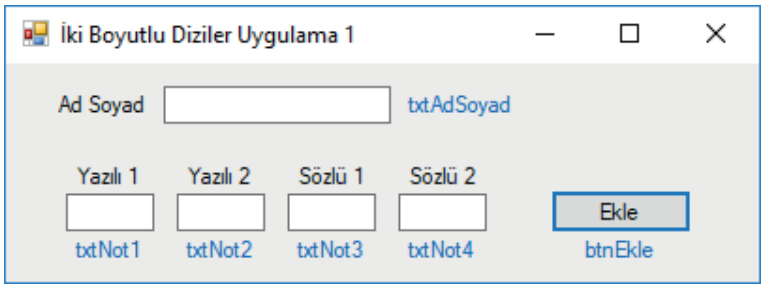
Görsel 4.14: İki boyutlu diziye değer aktarma

6. Uygulama

İki Boyutlu Diziler

Bu uygulamada üç öğrenciye ait 4 adet ders notu girişi yapılacaktır. Öğrenci adları tek boyutlu diziye, öğrenciye ait ders notları iki boyutlu diziye aktarılacaktır.

1. Adım: Görsel 4.15'te verilen form tasarımını yapınız. Form üzerindeki kontrollerin name değerlerini veriniz.



Görsel 4.15: İki boyutlu dizi form tasarımı uygulaması

2. Adım: Öğrenci adı soyadı değerlerinde tek boyutlu dizi, not değerlerinde iki boyutlu dizi kullanılacağı için dizi tanımlamalarını global olarak yapınız.

```
string[] isimler = new string[3]; //Tek boyutlu global dizi
int[,] notlar = new int[3,4];     //İki boyutlu global dizi
int index = 0;                    //Global değişken
```

3. Adım: Ekle butonu Click olayında TextBox'lara girilen değerleri ilgili dizilere aktarma işlemini gerçekleştiriniz.

```
private void btnEkle_Click(object sender, EventArgs e)
{
  isimler[index] = txtAdSoyad.Text;
  notlar[index, 0] = int.Parse(txtNot1.Text);
  notlar[index, 1] = int.Parse(txtNot2.Text);
  notlar[index, 2] = int.Parse(txtNot3.Text);
  notlar[index, 3] = int.Parse(txtNot4.Text);
  index++;
}
```

Sıra Sizde

1. Görsel 4.15'teki uygulamanın çalışma mantığını açıklayınız.
2. Görsel 4.15'teki uygulamada notları girilecek öğrenci sayılarını form içindeki bir TextBox'tan alacak şekilde kodları düzenleyiniz.

4.1.10. İki Boyutlu Dizi Elemanlarına Erişim

Bir boyutlu dizilerde olduğu gibi iki boyutlu dizi elemanlarına erişmek için de index numaraları kullanılır. Erişim sağlanan dizi elemanı; değişkenlere aktarma, hesaplamalar yapma, ekrana yazdırma, nesneye aktarma gibi işlemlerde kullanılır.

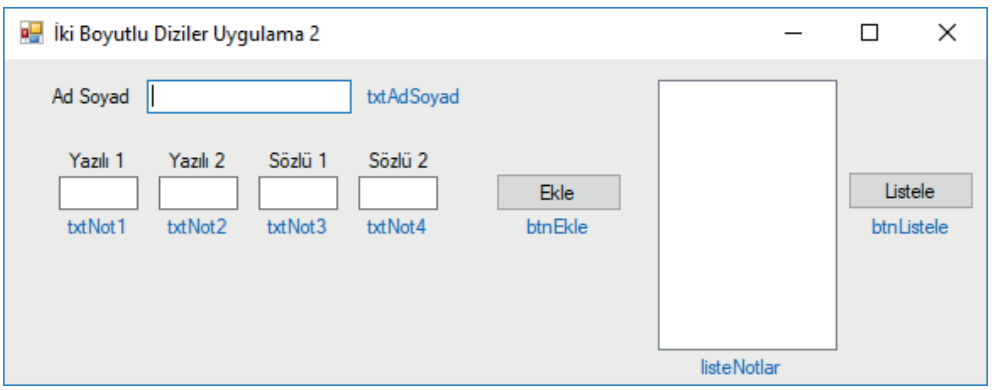
```
double[,] notlar = new double[3,4] {{45,55,60,65},{75,80,85,90},{10,20,30,40}};
// 3x4 elemanlı iki boyutlu dizi tanımlaması yapıldı. Aynı satırda değerler verildi.

int sayi1 = notlar[0,1];
// Dizinin 0,1 index numaralı değeri sayi1 ismindeki integer değişkenine aktarıldı.

int toplam = notlar[1,0] + notlar[1,1] + notlar[1,2] + notlar[1,3];
// Dizisinin 1. index numaralı satırındaki elemanlarla toplama yapıldı.

Console.WriteLine(notlar[2,1]);
// Dizisinin 2,1 index numaralı değeri Console ekranına yazdırıldı.

notlar[2,0] = notlar[1,3];
// Dizisinin 1,3 index numaralı değeri 2,0 index numaralı alanına aktarıldı.

label1.Text = notlar[2,2].ToString();
// Dizinin 2,2 index numaralı değeri label1 nesnesinin text özelliğine aktarıldı.
```

7. Uygulama

İki Boyutlu Diziler

Altıncı uygulamada diziye aktarılan not bilgilerinin ortalaması hesaplanarak ListBox içinde gösterilecektir.

1. Adım: Altıncı uygulama formuna Görsel 4.16'da olduğu gibi ListBox ve Button kontrollerini ekleyiniz.



Görsel 4.16: İki boyutlu dizi form tasarımı uygulaması

2. Adım: Listele butonu Click olayında isimlerle birlikte girilen notların ortalamasını hesaplayarak ListBox içinde gösteren kodlamaları yapınız.

```
private void btnListele_Click(object sender, EventArgs e)
{
    double toplam;
    for (int x = 0; x < 3; x++)
    {
        toplam = 0;
        for (int y = 0; y < 4; y++)
        {
            toplam += notlar[x, y];
        }
        listeNotlar.Items.Add(isimler[x]+" => "+ toplam / 4);
    }
}
```

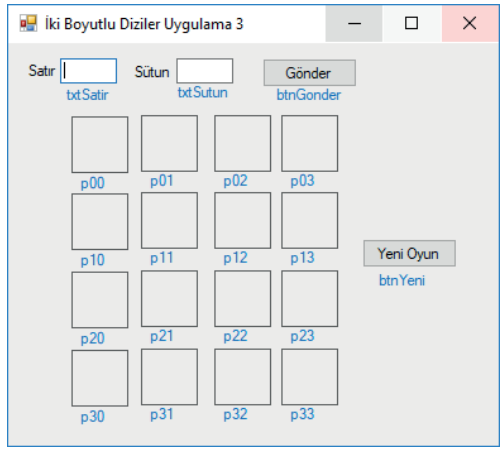
Sıra Sizde

İkinci adımda yazılan kodların çalışma mantığını açıklayınız.

8. Uygulama

İki Boyutlu Diziler

Bu uygulamada iki boyutlu diziler kullanılarak basit bir oyun programı yapılacaktır. Oyun programının amacı, 4 satır ve 4 sütundan oluşan iki boyutlu dizide rastgele bir konuma 1 değerini aktarmaktır. Dizinin geri kalan değerleri 0 olarak bırakılacaktır. Satır ve sütun bilgileri, form üzerinde bulunan iki adet TextBox'tan alınacaktır. Alınan satır ve sütun bilgileri dizinin satır ve sütun konumundaki değerle karşılaştırılarak dizi içindeki değer 1 ise oyun kaybedilecek ve ilgili PictureBox'ın rengi kırmızı olacak, dizi içindeki değer 1 değil ise ilgili PictureBox'ın rengi yeşil olacak şekilde kodlama yapınız.



Görsel 4.17: İki boyutlu dizi form tasarımı uygulaması

1. Adım: Yeni bir form oluşturarak 16 adet PictureBox, 2 adet TextBox ve 2 adet Button kontrolünü Görsel 4.17'deki gibi ekleyiniz. Kontrollere name değerlerini veriniz.

2. Adım: Byte tipinde 4 satır ve 4 sütundan oluşan iki boyutlu diziyi global olarak tanımlayınız.

```
byte[,] dizi = new byte[4, 4];
```

3. Adım: Yeni Oyun butonu Click olayında dizi içinde rastgele bir konuma 1 değerini aktarmak için Random sınıfından üretilen rastgele isminde bir nesne kullanınız. Random sınıfından üretilen bu nesne Next metodu ile birlikte 0 ile 4 arasında, 4 dâhil olmayacak şekilde bir sayı verir. Üretilen bu sayılar,

dizinin herhangi bir konumuna 1 değerini aktarmak için kullanılır. Dizinin belirtilen konumundaki sayı 1 olurken diğer konumlardaki değerler 0 olarak kalır.

```
private void btnYeni_Click(object sender, EventArgs e)
{
  Random rastgele = new Random(); // rastgele isminde bir Random nesnesi oluşturdu.
  int satirRastgele = rastgele.Next(4);// 0-4 arası (4 dâhil değil) üretilen sayıdır.
  int sutunRastgele = rastgele.Next(4); // 0-4 arası (4 dâhil değil) üretilen sayıdır.
  dizi[satirRastgele, sutunRastgele] = 1; // Dizi içinde rastgele bir konuma 1 değeri aktarıldı.
}
```

4. Adım: Gönder butonu Click olayında satır ve sütun TextBox'larına girilen bilgileri satir ve sutun isimli değişkenlere aktarınız. Bu değişkenleri kullanarak önce hangi PictureBox'ın seçileceğini belirleyiniz. Örneğin satır değeri 1, sütun değeri 3 ise formdaki PictureBox kontrollerinden name değeri "p13" seçilecektir. Dizi içinde [1,3] konumundaki değer 1 ise kutu rengi kırmızı, değilse kutu rengi yeşil olur.

```
private void btnGonder_Click(object sender, EventArgs e)
{
 byte satir = byte.Parse(txtSatir.Text);
 byte sutun = byte.Parse(txtSutun.Text);
 PictureBox kutu=this.Controls.Find("p"+satir+sutun, true)[0]as PictureBox;
// Controls.Find fonksiyonu name değerlerine göre  form içindeki kontrolleri bulmak için kullanılır.
// Bulunan kontroller as operatörü ile  PictureBox nesnelerine dönüştürülür.
 byte durum = dizi[satir, sutun];
 if (durum == 0)
   {kutu.BackColor = Color.Green;  }
   else
   {kutu.BackColor = Color.Red;  }
}
```

9. Uygulama

İki Boyutlu Diziler

Bu uygulamada satır ve sütun sayıları kullanıcı tarafından verilen iki boyutlu bir diziye form üzerinden girilen belirli aralıktaki rastgele sayıları aktarma işlemi yapılacaktır. Sonraki işlem, bu dizinin değerlerini ListBox nesnesi içinde göstermektir.

1. Adım: Görsel 4.18'deki form tasarımını yapınız ve form üzerindeki kontrollerin name değerlerini veriniz.

Görsel 4.18: İki boyutlu dizi form tasarımı uygulaması-4

2. Adım: Dizinin satır ve sütun sayıları birden çok yerde kullanılacağı için diziyi global olarak tanımlayınız.

```
int[,] dizi;
  int satirSayisi;
  int sutunSayisi;
```

3. Adım: Dizi Oluştur butonu Click olayında formdan gelen satır ve sütun bilgileri ile bir dizi oluşturunuz. Oluşturulan bu diziye aktarılacak rastgele sayıların hangi aralıkta olacağı bilgisi kullanılarak iki boyutlu bir dizi elde edilecektir. Bu dizinin her elemanına içi içe döngü kullanılarak rastgele sayılar atanacaktır.

```
private void btnDiziOlustur_Click(object sender, EventArgs e)
{
   int rastgeleMin= int.Parse(txtRastgeleMin.Text);
   int rastgeleMax = int.Parse(txtRastgeleMax.Text);
   satirSayisi = int.Parse(txtSatirSayisi.Text);
   sutunSayisi = int.Parse(txtSutunSayisi.Text);
   dizi = new int[satirSayisi, sutunSayisi];
   Random rastgele = new Random();
        for (int x = 0; x < satirSayisi; x++)
         {
                  for (int y = 0; y < sutunSayisi; y++)
                 {
                           dizi[x, y] = rastgele.Next(rastgeleMin, rastgeleMax);
                 }
           }
}
```

4. Adım: Dizi Göster butonu Click olayında rastgele sayılarla oluşturulan dizinin ListBox nesnesi içinde ait olduğu satır ve sütun bilgisi ile satır ve sütuna ait değerleri listeleyiniz.

```
private void btnlistele_Click(object sender, EventArgs e)
{
   for (int x = 0; x < satirSayisi; x++)
   {
      for (int y = 0; y < sutunSayisi; y++)
      {
        listeDizi.Items.Add(x+","+y+" => "+dizi[x,y]);
      }
    }
}
```

Sıra Sizde

1. Aynı uygulamayı üç boyutlu dizi kullanarak yeniden düzenleyiniz.
2. Üç boyutlu dizi oluşturmak ve listelemek için kaç adet iç içe döngü kullanmak gereklidir? Neden?

4.2. KOLEKSİYONLAR

Diziler, programlama dillerinde çokça kullanılan yapılardır. Dizilerde çok sayıda değer tutulabilir ve onlara erişim sağlanabilir ancak dizilerle işlem yapılırken iki sınırlama ile karşılaşılır. Bu sınırlamalar şunlardır:

- Dizilere aktarılacak değerler dizi ile aynı tipte olmalıdır.
- Dizilerin eleman sayıları önceden belirlenmelidir.

Programlamada bazen aynı veri tipine sahip olmayan değerlerle işlem yapmak zorunda kalınabilir. Nesne Tabanlı Programlama bu durumda koleksiyonları (collections) sunar. Koleksiyonları kullanabilmek için projeye **System.Collections** isim uzayı (namespace) dâhil edilmelidir. Koleksiyonların isim uzayına eklenmesi aşağıda verilmiştir.

```
using System.Collections;
```

Dizi ve koleksiyon arasındaki temel farklar Tablo 4.1'de belirtilmiştir.

Tablo 4.1: Dizi ve Koleksiyon Karşılaştırması

Diziler	Koleksiyonlar
Diziler aynı tip verileri saklar.	Koleksiyonlar aynı veya farklı tipteki verileri saklar.
Dizilerin eleman sayısı başlangıçta belirlenir ve sonradan değiştirilemez.	Koleksiyonların eleman sayılarını başlangıçta belirtmeye gerek yoktur.
Dizilerin eleman sayıları sabittir. Bu nedenle ihtiyaç durumunda eleman sayılarının artırılmasına ve azaltılmasına izin vermez.	Koleksiyonların eleman sayıları değiştirilebilir. İhtiyaca göre eleman sayıları artırılabilir ve azaltılabilir.
Diziler performans açısından koleksiyonlardan daha hızlıdır.	Koleksiyonlar performans açısından dizilerden daha yavaştır.
Bellek açısından dizilerin kullanılması tavsiye edilmez.	Bellek açısından koleksiyonların kullanılması tavsiye edilir.
Dizilerin elemanları üzerinde işlem yapmak için koleksiyonlardan daha az hazır metodu vardır.	Koleksiyonların elemanları üzerinde işlem yapmak için dizilerden daha fazla hazır metodu vardır.

Kodlama yaparken özel isteğe uyarlanmış koleksiyon oluşturulabileceği gibi Nesne Tabanlı Programlama dili ile gelen hazır koleksiyonlar da kullanılabilir. Hazır koleksiyonlardan bazıları şunlardır:

- ArrayList
- List
- Queue-Stack
- Dictionary
- HashTable
- SortedList

4.2.1. Boxing (Kutulama)-Unboxing (Kutu Açma)

Nesne Tabanlı Programlama dilinde **Value Type** (değer tipi) ve **Reference Type** (referans tipi) olmak üzere iki veri tipi vardır. Değer tipleri; int, char, byte, double vb. belleğinin **Stack** adı verilen kısmında tutulur. Referans tipleri ise object, string, class vb. belleğin **Heap** adı verilen kısmında tutulur.

Boxing, herhangi value (değer) tipideki değişkenin object (nesne) tipindeki değişkene dönüştürülmesidir.

```
int x = 1234;
object obj;
obj = x;
```

Unboxing, object (nesne) tipindeki değişkenin value (değer) tipindeki değişkene dönüştürülmesidir. Burada dikkat edilmesi gereken nokta, dönüştürülmek istenen nesne hangi tipte boxing yapılmış ise aynı tipte **Casting** yapılmalıdır. Aksi durumda veri tipleri aynı olmadığı için dönüştürme hatası (InvalidCastException) alınacaktır.

```
int y = (int)obj;
```

4.2.2. ArrayList Koleksiyonu

ArrayList, dizilerde olduğu gibi veri saklama amacıyla kullanılan bir koleksiyondur. Eleman sayıları dinamik olarak değişir ve farklı tiplerde veri saklama imkânı sunar. ArrayList elemanlarına erişim için index numaraları kullanılır. ArrayList tanımlanırken eleman sayısını belirtmeye gerek yoktur. Dinamik yapısı sayesinde kodlama sırasında veya çalışma anında ekleme, silme, araya ekleme, değerleri değiştirme gibi işlemler yapılabilir. ArrayList içine eklenen elemanlar object tipinde olacağı için veri eklerken boxing (kutulama), veri alırken de unboxing (kutu açma) işlemi yapılır.

Koleksiyon nesnesi ArrayList'in oluşturulması aşağıda verilmiştir. Komut derlendiğinde **ArrayList** sınıfından **liste** adında bir nesne üretilir.

```
ArrayList liste = new ArrayList();
```

ArrayList Veri Ekleme

Koleksiyonlara veri eklemek için **Add** metodu kullanılır. Koleksiyonlar da diziler gibi index numaralarına sahiptir. ArrayList'e ilk eklenen elemanın index numarası 0 olur, diğer elemanlar da 0'dan başlayarak sıralanır.

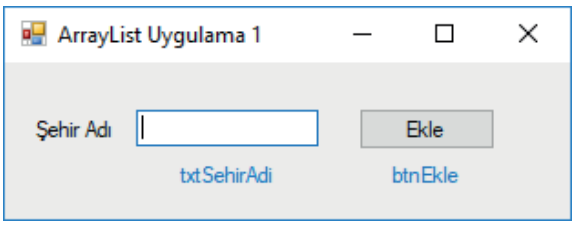
```
ArrayList liste = new ArrayList();
liste.Add("Bilişim"); // Metinsel
liste.Add(100);       // Tam sayı
liste.Add('m');       // Karakter
liste.Add(3.14);      // Ondalık sayı
liste.Add(true);      // Mantıksal
```

10. Uygulama

ArrayList

Şehir isimlerinin saklanacağı bir ArrayList oluşturarak formdan girilen şehir isimlerini tanımlanan ArrayList içine ekleme işlemi yapılacaktır.

1. Adım: Görsel 4.19'da verilen form tasarımını yapınız ve form içindeki kontrollere name değerlerini veriniz.



Görsel 4.19: ArrayList form tasarımı uygulaması

2. Adım: Ekle butonu Click olayında global olarak tanımlanan sehirler ismindeki ArrayList'e TextBox'tan gelen verileri aktarınız.

```
ArrayList sehirler = new ArrayList();
private void btnEkle_Click(object sender, EventArgs e)
{
  sehirler.Add(txtSehirAdi.Text);
}
```

ArrayList Elemanlarına Erişim

ArrrayList içindeki elemanlara erişim için dizilerde olduğu gibi index numaraları kullanılır.

```
sehirler[0]  // 0 index numaralı ilk eleman
sehirler[1]  // 1 index numaralı ikinci eleman
sehirler[2]  // 2 index numaralı üçüncü eleman
```

Örnekte gösterildiği gibi index numaraları 0'dan başlanarak eleman sayısının 1 eksiğine kadar istenilen değerlere erişim sağlanabilir. Dizilerde olduğu gibi ArrayList elemanlarına sınırları dışında bir index numarası ile ulaşılmaya çalışıldığında hata ile karşılaşılır.

```
sehirler[1]="Mardin"; // 1 index numaralı elemana erişim sağlanarak değeri değiştirildi.
```

```
Label1.Text=(string)sehirler[1];   // 1 index numaralı elemana erişim sağlanarak unboxing işlemiyle
// Label kontrolü içinde gösterilmiştir.
```

11. Uygulama

ArrayList

Onuncu uygulama biraz daha geliştirilerek ArrayList nesnesine eklenen veriler index numaraları ile birlikte ListBox kontrolü içinde gösterilecektir.

1. Adım: Görsel 4.20'de verilen form tasarımını yapınız ve form içindeki kontrollere name değerlerini veriniz.

Görsel 4.20: ArrayList form tasarımı uygulaması

2. Adım: Listele butonu Click olayında aşağıdaki kodları yazarak listeleme işlemini gerçekleştiriniz.

```
private void btnListe_Click(object sender, EventArgs e)
{
   for (int i = 0; i < sehirler.Count; i++)
   {
      lbListe.Items.Add(sehirler[i]);
   }
}
```

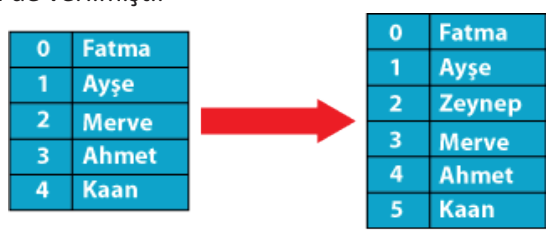
Uygulamada ArrayList koleksiyonu içindeki değerlere ulaşmak için bir döngü kullanıldı. Bu döngüyle 0'dan başlanarak koleksiyonun eleman sayısına kadar dönme işlemi yapıldı. Koleksiyonlarda eleman sayısı Count özelliği ile alınıp döngü değişkeni koleksiyon index numarası olarak kullanıldı ve her elemana erişim sağlandı.

Sıra Sizde

1. İkinci adımdaki for döngüsü yerine foreach döngüsünü kullanarak aynı işlemleri yapınız.
2. İkinci adımda koleksiyon değerlerinin ListBox içinde gösterilmesinde niçin unboxing yapılmamıştır? Açıklayınız.

Insert metodu, koleksiyonlara değer eklemenin bir diğer yoludur. Bu metodun add metodundan farkı, eklenecek değerin hangi sıraya atanacağının belirtilmesidir. Insert metodu kullanımından sonra koleksiyonun bellek üzerindeki durumu Görsel 4.21'de verilmiştir.

```
ArrayList isimler = new ArrayList();
isimler.Add("Fatma");
isimler.Add("Ayşe");
isimler.Add("Merve");
isimler.Add("Ahmet");
isimler.Add("Kaan");
isimler.Insert(2,"Zeynep");
```

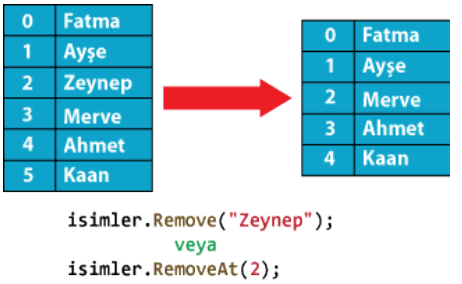


Görsel 4.21: ArrayList insert metodu kullanımı

ArrayList Veri Silme

ArrayList'lerde bir veriyi silmek için **Remove** ve **RemoveAt** metotları vardır. **Remove** metodunda silinecek nesnenin değeri, **RemoveAt** metodunda ise silinecek nesnenin index numarası kullanılır. Veri silme metotlarının kullanımından sonra koleksiyonun bellek üzerindeki durumu Görsel 4.22'de verilmiştir.

ArrayList içindeki tüm verileri silmek için **Clear()** metodu kullanılır.




```
isimler.Remove("Zeynep");
        veya
isimler.RemoveAt(2);
```

Görsel 4.22: ArrayList Remove ve RemoveAt metodu kullanımı

Sıra Sizde

ArrayList içinde aynı değerlere sahip birden çok eleman varsa Remove metodu aynı değerlere sahip elemanlardan ilk bulunanı siler. Diğer elemanları da silmek için ne yapılmalıdır?

ArrayList Veri Arama

Bir ArrayList içinde bir verinin aranması için birkaç yöntem bulunur. Bunlardan bir tanesi **Contains** metodudur. Contains metodu; aranan veri koleksiyonda varsa **true**, yoksa **false** değerini geriye döndürür.

```
if(isimler.Contains("Ahmet"))
{
  label1.Text = "Aranan veri bulundu.";
}
else
{
  label1.Text = "Bulunamadı.";
}
```

ArrayList içinde bir diğer arama yöntemi **IndexOf** metodudur. Bu metot, **Contains** metodundan farklı olarak koleksiyon içinde aranan veri bulunursa index numarasını, bulunamazsa **-1** değerini geriye döndürür.

```
int durum = isimler.IndexOf("Ahmet");
if (durum != -1)
{
    label1.Text = durum+ "    index numaralı aranan veri bulundu.";
}
else
{
    label1.Text = "Bulunamadı.";
}
```

12. Uygulama

Uygulamada sehirler isminde global tanımlanan bir ArrayList üzerinde ekleme, araya ekleme, güncelleme, silme ve arama işlemleri yapılacaktır.

1. Adım: Görsel 4.23'teki form tasarımını yapınız ve form içindeki kontrollere name değerlerini veriniz.

Görsel 4.23: ArrayList form tasarımı uygulaması

2. Adım: Ekle butonu Click olayında global olarak tanımlanan sehirler ismindeki ArrayList'e TextBox'tan gelen verileri aktarınız.

```
ArrayList sehirler = new ArrayList();
private void btnEkle_Click(object sender, EventArgs e)
{
  sehirler.Add(txtSehirler.Text);
  Listele();
}
```

3. Adım: Ekle butonu içinde Listele isminde bir metot kullanınız. Bu metodun görevi; eklenen, silinen veya değeri değiştirilen ArrayList elemanlarını ListBox kontrolü içinde göstermektir.

```
private void Listele()
{
  listeSehirler.Items.Clear();
  foreach (string sehir in sehirler)
  {
     listeSehirler.Items.Add(sehir);
  }
}
```

4. Adım: ListBox içindeki elemanın index numarası, ArrayList elemanları ile aynıdır. Araya Ekle butonu Click olayında ListBox içinden seçilen elemanın index numarası, SelectedIndex özelliği kullanılarak bir değişkene aktarılır. Bu değişkeni insert metodu içinde kullanarak araya ekleme işlemini gerçekleştiriniz.

```
private void btnAraEkle_Click(object sender, EventArgs e)
{
   int indexNo = listeSehirler.SelectedIndex;
   sehirler.Insert(indexNo, txtSehirler.Text);
   Listele();
}
```

5. Adım: Güncelle butonu Click olayında ListBox içinden seçilen elemanın index numarasını kullanarak ArrayList elemanlarının değerlerini değiştiriniz.

```
private void btnGuncelle_Click(object sender, EventArgs e)
{
   int indexNo = listeSehirler.SelectedIndex;
   sehirler[indexNo] = txtSehirler.Text;
   Listele();
}
```

6. Adım: Sil butonu Click olayında ListBox üzerinden index numarası alınan ArrayList elemanını silme işlemini yapınız.

```
private void btnSil_Click(object sender, EventArgs e)
{
  int indexNo = listeSehirler.SelectedIndex;
  sehirler.RemoveAt(indexNo);
  Listele();
}
```

7. Adım: Ara butonu Click olayında TextBox'a yazılan değeri ArrayList içinde arama işlemini yapınız. Aranan değer bulunursa labelDurum'a "Aranan Değer Bulundu." mesajı verilecek ve IndexOf metodu kullanılarak ListBox içindeki konumuna gidecektir. Aranan değer bulunamazsa "Aranan Değer Bulunamadı." mesajı verilecektir.

```
private void btnAra_Click(object sender, EventArgs e)
{
   if (sehirler.Contains(txtSehirler.Text))
   {
      labelDurum.Text = "Aranan Değer Bulundu.";
      listeSehirler.SelectedIndex = sehirler.IndexOf(txtSehirler.Text);
   }
   else
   {
       labelDurum.Text = "Aranan Değer Bulunamadı.";
   }
}
```

ArrayList Veri Sıralama

ArrayList içindeki veriler, eklenme sırasına göre 0 index numarasından başlayarak devam eder. **Reverse** metodu bu sıralamayı tamamen tersine çevirir.

```
sehirler.Reverse();
```

Bir diğer sıralama ise **Sort** metodudur. Sort metodu, ArrayList içindeki değerleri artan bir sıra hâlinde yeniden düzenler. Sort metodu; koleksiyondaki elemanlar yazı tipinde ise a'dan z'ye, sayısal tipte ise küçükten büyüğe doğru sıralar.

```
sehirler.Sort();
```

Sıra Sizde

On ikinci uygulamadaki forma iki adet buton ekleyiniz. Butonlardan birinin görevi sıralama, diğerinin görevi ise tersten sıralama olsun. Bu butonlara tıklandığında ListBox içinde ArrayList'in sıralanmış veya tersten sıralanmış hâllerini gösteriniz.

4.2.3. List Koleksiyonu

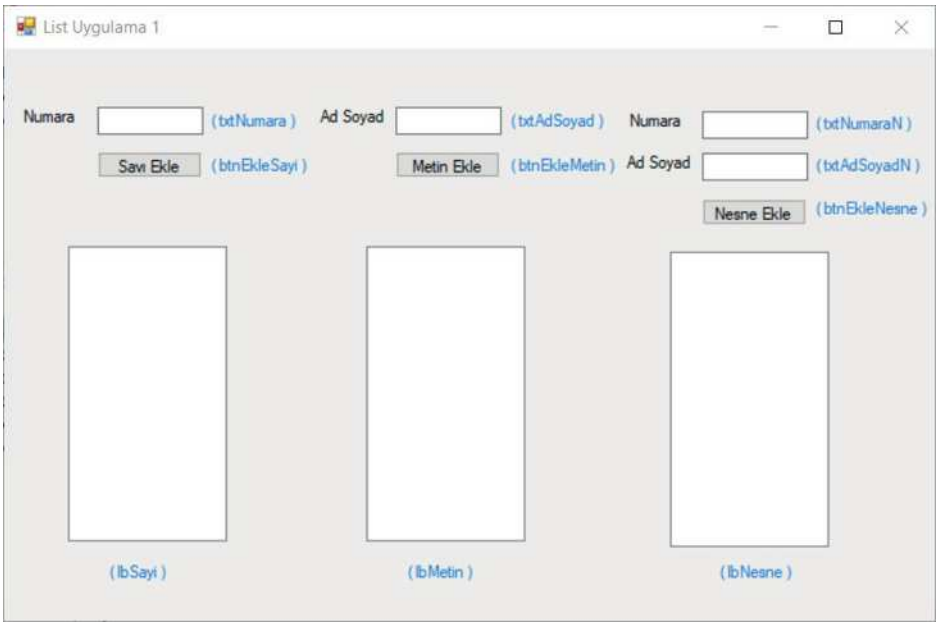
List koleksiyonu ile ArrayList koleksiyonu benzerlik gösterir. Aralarındaki fark; List koleksiyonunun generic, ArrayList koleksiyonunun ise non-generic yapıda olmasıdır. Generic koleksiyonlarda mutlaka içinde saklanacak verinin tipi belirtilmelidir çünkü belirlenen tipin dışında veri saklanmaya çalışıldığında hata ile karşılaşılır. Non-generic koleksiyonlarda ise bu işlem yapılmaz. Non-generic koleksiyonların veri tipi obje olarak belirlenir ve bu koleksiyonlarda her tipten veri saklanabilir. Generic koleksiyonlarda boxing-unboxing işlemleri yapılmazken non-generic koleksiyonlarda yapılır.

13. Uygulama

http://kitap.eba.gov.tr/KodSor.php?KOD=21080

Bu uygulamada List koleksiyonuna sayısal ifadeler ve metin ifadeleri nesne ekleme yöntemiyle eklenecektir.

1. Adım: Görsel 4.24'teki gibi form tasarımını yapınız ve form içindeki kontrollere name değerlerini veriniz.



Görsel 4.24: List form tasarımı uygulaması

2. Adım: Uygulamaya Ogrenciler adında bir sınıf oluşturarak özelliklerini veriniz.

```
class Ogrenciler
{
  public int Numara { get; set; }
  public string AdSoyad { get; set; }
}
```

3. Adım: Oluşturulan forma global olarak üç adet List koleksiyonu oluşturunuz.

```
List<int> numaralarList = new List<int>();
List<string> adsoyadList = new List<string>();
List<Ogrenciler> ogrencilerList = new List<Ogrenciler>();
```

4. Adım: Sayı Ekle butonu Click olayına Numara TextBox'ı içine girilen değerleri List koleksiyonuna ekleme ve ListBox içinde gösterme işlemini yapınız.

```
private void btnEkleSayi_Click(object sender, EventArgs e)
{
    numaralarList.Add(int.Parse(txtNumara.Text));
    lbSayi.Items.Clear();
    foreach (var item in numaralarList)
     {
        lbSayi.Items.Add(item);
     }
}
```

5. Adım: Metin Ekle butonu Click olayına Ad Soyad TextBox'ı içine girilen değerleri List koleksiyonuna ekleme ve ListBox içinde gösterme işlemini yapınız.

```
private void btnEkleSayi_Click(object sender, EventArgs e)
{
    numaralarList.Add(int.Parse(txtNumara.Text));
    lbSayi.Items.Clear();
    foreach (var item in numaralarList)
     {
        lbSayi.Items.Add(item);
     }
}
```

6. Adım: Nesne Ekle butonu Click olayına Numara ve Ad Soyad TextBox'ı içine girilen değerleri nesne yoluyla List koleksiyonuna ekleme ve ListBox içinde gösterme işlemini yapınız.

```
private void btnEkleNesne_Click(object sender, EventArgs e)
{
   Ogrenciler ogrenci = new Ogrenciler();
   ogrenci.Numara = int.Parse(txtNumaraN.Text);
   ogrenci.AdSoyad = txtAdSoyadN.Text;
   ogrencilerList.Add(ogrenci);
   lbNesne.Items.Clear();
   foreach (var item in ogrencilerList)
   {
      lbNesne.Items.Add(item.Numara+" "+item.AdSoyad);
   }
}
```

Sıra Sizde

ArrayList ile List arasındaki işlem süresi sonuçlarını karşılaştırıp yorumlayınız.

4.2.4. Queue-Stack Koleksiyonları

Queue kelimesi kuyruk anlamına gelir. İlk giren eleman ilk çıkar işleyişine sahip bir koleksiyondur (FIFO-First In First Out). Koleksiyondan bir eleman çıkarılmak istendiğinde kuyruğa ilk eklenen eleman çıkartılacaktır. Yeni eklenecek eleman ise kuyruğun en sonuna getirilir.

Aslında bu yapıyla günlük hayatta çokça karşılaşılır. Örneğin sıra numarasına göre işlem yapılan bir işletmede önce bir sıra numarası alınır. Alınan bu sıra numarası kuyruğun en sonundadır. İşlem yapılan numaralar sıradan çıkar ve işlem sırası diğer numaralara gelir.

Queue ekleme ve çıkarma işleminin iki metodu vardır:

1. Enqueue() metodu kuyruğun sonuna bir eleman ekler.

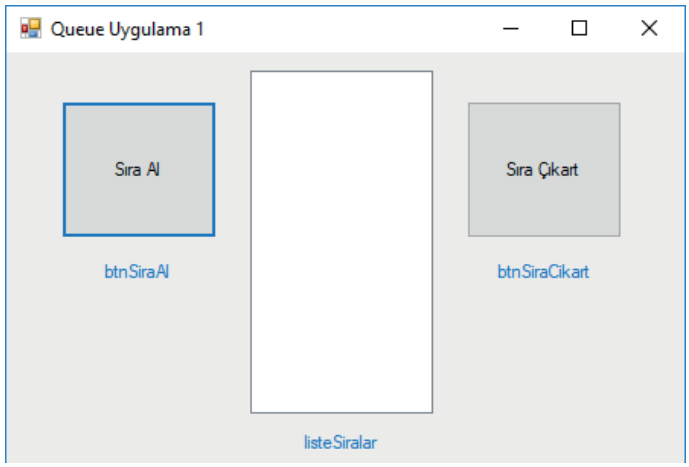
2. Dequeue() metodu kuyruğun başındaki elemanı çıkarır.

14. Uygulama

Queue Koleksiyonu

Uygulamada Queue sınıfından üretilen nesne ile iki Button bir ListBox kontrolü kullanılarak kuyruğa alma ve kuyruktan çıkarma işlemleri yapılacaktır.

1. Adım: Görsel 4.25'teki form tasarımını yapınız ve form üzerindeki kontrollerin name değerlerini veriniz.



Görsel 4.25: Queue form tasarımı uygulaması

2. Adım: Queue sınıfından üretilecek nesneyi ve bu nesneye aktarılacak sıra numaraları için global değişkenleri oluşturunuz.

```
Queue kuyruk = new Queue();
int sira = 0;
```

3. Adım: Sıra Al butonu Click olayında sira değişkeninin değerini bir artırarak oluşturulan Queue nesnesine aktarınız.

```
private void btnSiraAl_Click(object sender, EventArgs e)
{
  sira++;
  kuyruk.Enqueue(sira);
  Listele();
}
```

4. Adım: Queue nesnesindeki değerleri ListBox içinde göstermek için Listele adında bir metot oluşturarak sıra durumunu bu metot içinde gösteriniz.

```
private void Listele()
{
  listeSiralar.Items.Clear();
  foreach (int sira in kuyruk)
  {
    listeSiralar.Items.Add(sira);
  }
}
```

5. Adım: Sıra Çıkart butonu Click olayında oluşturduğunuz Queue nesnesine verilen ilk değeri kuyruktan çıkarma işlemini gerçekleştiriniz. Listele metodunu kullanarak en güncel kuyruk değerlerini ListBox içinde gösteriniz.

```
private void btnSiraCikart_Click(object sender, EventArgs e)
{
  kuyruk.Dequeue();
  Listele();
}
```

Sıra Sizde

Beşinci adımda kuyrukta hiç sıra yokken çıkartma işlemi yapıldığında nasıl bir hata ile karşılaşılır? Bu hatanın çözümü için ne yapılabilir? Önerilerinizi sınıfınızla paylaşınız.

Stack kelimesi yığın anlamına gelir. Son giren eleman ilk çıkar işleyişine sahip bir koleksiyondur (LIFO-Last In First Out). Koleksiyondan bir eleman çıkarılmak istendiğinde yığına son eklenen eleman çıkartılacaktır. Örneğin ofis uygulamalarında yapılan işlemler uygulama tarafından hafızaya alınır. Uygulamada geri alma işlemi yapıldığında en son işlem gerçekleştirilir. Geri alma işlemine devam edildiğinde sondan başa doğru hafızadaki işlemler gerçekleştirilir.

Stack içine ekleme ve çıkarma işleminin iki metodu vardır:

1. Push() metodu Stack içine bir değer ekler.

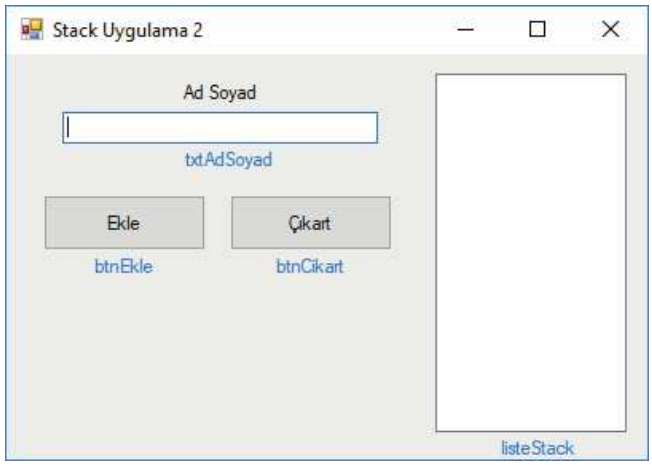
2. Pop() metodu Stack içine eklenen son değeri çıkarır.

15. Uygulama

Stack Uygulama

Uygulamada Stack sınıfından üretilen nesne ile bir TexBox, iki Button, bir ListBox kontrolü kullanarak yığına alma ve yığından çıkarma işlemleri yapılacaktır.

1. Adım: Görsel 4.26'daki form tasarımını yapınız ve form üzerindeki kontrollerin name değerlerini veriniz.



Görsel 4.26: Stack form tasarımı uygulaması-2

2. Adım: Stack sınıfından üretilecek nesneyi global olarak oluşturunuz.

```
Stack yigin = new Stack();
```

3. Adım: Ekle butonu Click olayında TextBox içinden alınan değerleri Stack içine aktarınız.

```
private void btnEkle_Click(object sender, EventArgs e)
{
  yigin.Push(txtAdSoyad.Text);
  Listele();
}
```

4. Adım: Stack içindeki değerleri ListBox içinde göstermek için Listele adında bir metot oluşturarak listeleme işlemlerini yapınız.

```
private void Listele()
{
  listeStack.Items.Clear();
  foreach (string eleman in yigin)
  {
    listeStack.Items.Add(eleman);
  }
}
```

5. Adım: Çıkart butonu Click olayında Stack içinden son değeri çıkarınız. Listele metodunu kullanarak en güncel Stack değerlerini ListBox içinde gösteriniz.

```
private void btnCikart_Click(object sender, EventArgs e)
{
    yigin.Pop();
    Listele();
}
```

4.2.5. Dictionary Koleksiyonu

Dizi ve ArrayList içine eklenen elemanlar bellek üzerine sıralı olarak yerleşir. Dizi ile ArrayList elemanlarına 0'dan başlanarak index numarası verilir ve elemanlara erişim için bu index numaraları kullanılır. Dictionary koleksiyonunda index-değer ilişkisi Key (Anahtar) - Value (Değer) olarak kullanılır. Key, dizilerdeki index numarası; Value ise index numarası ile belirtilen değer olarak düşünülebilir. Key; benzersiz olmak şartıyla int, string, byte, object vb. olabilir.

Dictionary koleksiyonunun kullanımı aşağıda verilmiştir.

```
Dictionary<KeyTipi,ValueTipi> koleksiyon_adı = new Dictionary<KeyTipi,ValueTipi>();
```

Kodlamalarda Dictionary nesnesi tanımlaması ve değer aktarımında Key-Value tiplerinin farklı kullanımları aşağıda gösterilmiştir.

```
Dictionary<int,string> sehirler = new Dictionary<int,string>();
sehirler.Add(1,"Adana");
sehirler.Add(2,"Adıyaman");
```

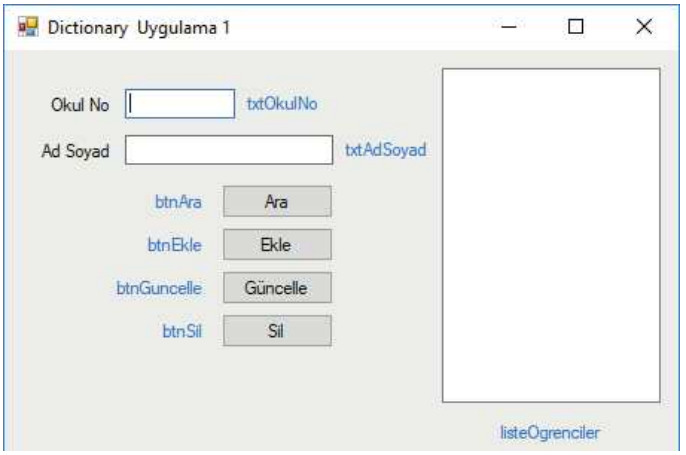
```
Dictionary<string, int> sehirler = new Dictionary<string, int>();
sehirler.Add("Adana", 1);
sehirler.Add("Adıyaman", 2);
```

16. Uygulama

Dictionary Koleksiyonu

Bu uygulamada okul öğrencilerinin numaralarını ve isimlerini bir Dictionary koleksiyonuna ekleme, silme, güncelleme ve arama işlemleri gerçekleştirilecektir.

1. Adım: Görsel 4.27'deki form tasarımını yapınız ve form üzerindeki kontrollerin name değerlerini veriniz.



Görsel 4.27: Dictionary form tasarımı uygulaması

2. Adım: Bu uygulamada Dictionary nesnesinin anahtarını okul numarası ve integer tipinde, değerini ise öğrencilerin adı soyadı ve string tipinde tanımlayınız.

```
Dictionary<int,string> ogrenciler = new Dictionary<int, string>();
int anahtar;
string deger;
```

3. Adım: Ekle butonu Click olayında okul numaralarını Key, öğrencilerin adı soyadını Value olarak koleksiyona ekleyiniz.

```
private void btnEkle_Click(object sender, EventArgs e)
{
  anahtar = int.Parse(txtOkulNo.Text);
  deger = txtAdSoyad.Text;
  ogrenciler.Add(anahtar,deger);
  Listele();
}
```

4. Adım: Dictionary koleksiyonundaki Key ve Value bilgilerini ListBox içinde göstermek için Listele isminde bir metot kullanınız. Bu metot, foreach döngüsü içinde ogrenciler koleksiyonundaki tüm Key ve Value bilgilerini var tipinde ismi ogrenci olan nesneye sırasıyla aktaracaktır. Bu metotla döngü her döndüğünde ogrenci nesnesini kullanarak ListBox içine Key ve Value bilgileri eklenecektir.

```
private void Listele()
{
   listeOgrenciler.Items.Clear();
   foreach (var ogrenci in ogrenciler)
   {
     listeOgrenciler.Items.Add(ogrenci.Key+"-"+ ogrenci.Value);
   }
}
```

5. Adım: Güncelle butonu Click olayında okul numarasına göre öğrenci isimlerini güncelleme işlemini gerçekleştiriniz.

```
private void btnGuncelle_Click(object sender, EventArgs e)
{
  anahtar = int.Parse(txtOkulNo.Text);
  deger = txtAdSoyad.Text;
  ogrenciler[anahtar] = deger;
  Listele();
}
```

6. Adım: Sil butonu Click olayında okul numarasına göre Dictionary koleksiyonundan öğrenci bilgilerini silme işlemini gerçekleştiriniz.

```
private void btnSil_Click(object sender, EventArgs e)
{
    anahtar = int.Parse(txtOkulNo.Text);
    ogrenciler.Remove(anahtar);
    Listele();
}
```

7. Adım: Ara butonu Click olayında okul numarasının veya öğrenci adının koleksiyon içinde bulunup bulunmadığını kontrol ediniz, okul numarası veya öğrenci adı varsa mesaj olarak gösterme işlemini yapınız.

```
private void btnAra_Click(object sender, EventArgs e)
{
    bool durum=false;
    if(txtOkulNo.Text!="")
    {
        anahtar = int.Parse(txtOkulNo.Text);
        durum = ogrenciler.ContainsKey(anahtar);
    }
    else
    {
        deger = txtAdSoyad.Text;
        durum = ogrenciler.ContainsValue(deger);
    }
    if(durum==true)
    {
        MessageBox.Show("Öğrenci Kayıtlıdır.");
    }
    else
    {
        MessageBox.Show("Öğrenci Kayıtlı Değildir.");
    }
}
```

Sıra Sizde

On altıncı uygulamada koleksiyon tanımlanırken öğrencinin adı soyadı Key olarak kullanılabilir mi? Açıklayınız.

4.2.6. Hashtable Koleksiyonu

Hashtable, non-generic bir koleksiyondur. Dictionary koleksiyonunda olduğu gibi Key-Value tipleri belirtilmez. Hashtable, karışık tablo anlamına gelir. Bu koleksiyona yalnızca bir defa kullanılmak şartıyla Key

değeri olarak istenilen herhangi bir değer atanabilir.

Hashtable koleksiyonunda kullanılan metot ve özellikler aşağıda verilmiştir.

Hashtable nesnesi oluşturma

```
Hashtable ogrenciler = new Hashtable();
```

Hashtable koleksiyonuna veri ekleme

```
ogrenciler.Add(368,"Ahmet");
ogrenciler.Add("Ahmet",368);
```

Hashtable koleksiyonundan veri silme

```
ogrenciler.Remove(368);
ogrenciler.Remove("Ahmet");
```

Hashtable koleksiyonunda veri güncelleme

```
ogrenciler[368] = "Mehmet";
```

Hashtable koleksiyonunda Key içeriğini listeleme

```
foreach (var anahtar in ogrenciler.Keys)
{
  Console.WriteLine(anahtar);
}
```

Hashtable koleksiyonunda Value içeriğini listeleme

```
foreach (var deger in ogrenciler.Values)
{
   Console.WriteLine(deger);
}
```

Hashtable koleksiyonu Key ve Value içeriğini listelemek için DictionaryEntry isimde bir sınıf kullanılarak koleksiyon içindeki elemanların hem Key hem de Value değerlerine erişim sağlanabilir.

```
foreach (DictionaryEntry eleman in ogrenciler)
{
   Console.WriteLine(eleman.Key + " - " + eleman.Value);
}
```

Sıra Sizde

On altıncı uygulamada hazırladığınız formu Hashtable koleksiyonunu kullanarak yeniden yapınız.

4.2.7. SortedList Koleksiyonu

SortedList koleksiyonu, Hashtable koleksiyonuna benzer. SortedList koleksiyonunun en önemli özelliği, içindeki verileri anahtarın içeriğine göre sıralamasıdır. Karışık olarak verilen anahtar değerlerini bile kendi içinde artan şekilde bir sıralama yaparak koleksiyon içine ekler.

SortedList koleksiyonunda kullanılan metot ve özellikler aşağıda verilmiştir.

SortedList nesnesi oluşturma

```
SortedList ogrenciler = new SortedList();
```

SortedList koleksiyonuna veri ekleme

```
ogrenciler.Add(368, "Ahmet");
```

SortedList koleksiyonundan veri silme

```
ogrenciler.Remove(368);
```

SortedList koleksiyonunda veri güncelleme

```
ogrenciler[368] = "Mehmet";
```

SortedList koleksiyonu Key ve Value içeriğini listelemek için DictionaryEntry isimde bir sınıf kullanılarak koleksiyon içindeki elemanların hem Key hem de Value değerlerine erişim sağlanabilir.

```
foreach (DictionaryEntry eleman in ogrenciler)
{
  Console.WriteLine(eleman.Key+" - "+eleman.Value);
}
```

Sıra Sizde

On altıncı uygulamada hazırladığınız formu SortedList koleksiyonunu kullanarak yeniden yapınız.

ÖLÇME VE DEĞERLENDİRME

A) Aşağıdaki sorularda yer alan boşlukları soru kökünde belirtildiği şekilde doldurunuz.

1. Aşağıdaki kod satırında bulunan boşluklara, integer tipte "numaralar" isminde bir dizi oluşturmak için uygun ifadeleri yazınız.

```
_____  _____=={10,20,30,40};
```

2. Aşağıdaki kod satırında bulunan boşluklara, string tipte "sehirler" isminde 81 elemanlı bir dizi oluşturmak için uygun ifadeleri yazınız.

```
_____  _____ = new _____
```

3. Aşağıda tanımlanan diziye göre x değişkenin değeri ne olur?

```
byte[ ]  sayilar  =  new byte[ ]{ 4,3,2,1};
byte x=sayilar[3];
```

B) Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

4. int[ ] sayilar = {10, 32, 60, 100, 90, 5}; şeklinde oluşturulan bir dizi için sayilar.Length() metodu çalıştırıldığında komutun döndürdüğü değer aşağıdakilerden hangisidir?

A) 5 B) 6 C) 7 D) 10 E) Hata verir.

5. Aşağıdakilerden hangisi 4 elemanlı integer tipinde tanımlanmış bir dizidir?

A) int[] dizi = new int[4]; B) int[4] dizi; C) int[] dizi = 4;

D) int dizi[4]; E) int[] dizi=new byte[4];

6. Dizilerle ilgili aşağıdaki ifadelerden hangisi doğrudur?

A) Dizi elemanları sadece integer tipinde olur.
B) Dizinin Rank özelliği dizideki eleman sayısını verir.
C) Dizinin Length özelliği dizinin boyutunu verir.
D) Sayısal dizilerde ön tanımlı değeri 0'dır.
E) Dizilere istenilen tipte değer verilebilir.

7. Aşağıdakilerden hangisi 4 satır ve 5 sütundan oluşan iki boyutlu dizi tanımlamasıdır?

A) int[4, 5] sayilar = new int[,];
B) int [4][5] sayilar = new int[];
C) int [,] sayilar = new int[4, 5];
D) int [,] sayilar = new int[5, 4];
E) int[5,4] sayilar = new int[];

8. Aşağıdakilerden hangisi 2 satır ve 3 sütunlu bir dizinin doğru tanımlanmış hâlidir?

A) int[,] sayi= new int[2, 3]{{1, 2},{10,11},{ 5, 6}};
B) int[,] sayi = new int[2, 3]{};
C) int[,] a = new int[1, 2];
D) int[,] a = new int[1, 2]{{7, 1, 9}, {2, 5, 6}};
E) int[,] sayi= new int[2, 3]{{7, 1, 9},{2, 5, 6}};

9.

```
int[] sayilar = { 1, 4, 8, 2 };
int toplam  = 0;
for ( int index=0; index < sayilar.Length; index++ )
{
 toplam = toplam + sayilar[index] ;
}
```

Yukarıda verilen kodlar çalıştırıldığında toplam değişkenin değeri aşağıdakilerden hangisidir?

A) 1482 B) 15 C) 13 D) 5 E) 0

10. Aşağıdakilerden hangisi bir ArrayList tanımlamasıdır?

A) ArrayList[] liste=new ArrayList();
B) ArrayList liste=new ArrayList();
C) ArrayList liste=ArrayList();
D) ArrayList liste=new ArrayList[10];
E) ArrayList() liste=new ArrayList();

11. Aşağıdakilerden hangisi bir ArrayList içine veri ekleme metodudur?

A) liste.Add(“abc”) B) liste.Set(“abc”) C) liste.Size(“abc”)
D) liste.Count(“abc”) E) liste.Get(“abc”)

12.

```
ArrayList liste = new ArrayList();
liste.add(“Ali”);
liste.add(“Elif”);
liste.add(“Can”);
liste.add(“Fatih”);
```

```
Ali
Elif
Mehmet
Can
Fatih
```

Yukarıda yer alan kodlarda koleksiyona eklenen değerlerden sonra yandaki listeye benzemesi için hangi işlem yapılmıştır?

A) liste.Add(“Mehmet”) B) liste.Add(2,”Mehmet”) C) liste.Add(“Mehmet”,2)
D) liste.Insert(“Mehmet”,2); E) liste.Insert(2,”Mehmet”)

13.

```
ArrayList liste = new ArrayList();
liste.add(“Ali”);
liste.add(“Elif”);
liste.add(“Can”);
liste.add(“Fatih”);
liste.Insert(0,”Mehmet”);
```

Yukarıda yer alan kodlamalara göre liste[2] değeri aşağıdakilerden hangisidir?

A) Ali B) Can C) Mehmet D) Elif E) Fatih

14.

```
ArrayList liste = new ArrayList();
liste.add("Ali");
liste.add("Elif");
liste.add("Can");
liste.add("Fatih");
liste.add("Mehmet");
```

Yukarıda yer alan kodlamalara göre aşağıdaki kodlardan hangisi Can ismini Ayşe olarak değiştirir?

A) liste("Can","Ayşe"); B) liste[2]="Ayşe"; C) liste["Can"] = "Ayşe";

D) liste(2)="Ayşe"; E) liste.Set("Can","Ayşe");

15.

```
ArrayList liste = new ArrayList();
liste.add("Ali");
liste.add("Elif");
liste.add("Can");
liste.add("Fatih");
liste.add("Mehmet");
```

Ali
Elif
Fatih
Mehmet

Yukarıda yer alan kodlamada koleksiyona eklenen değerlerden sonra yandaki listeye benzemesi için hangi işlem yapılmıştır?

A) liste.RemoveAt(2); B) liste.Remove(2,"Can"); C) liste.Remove("Can",2)

D) liste.Remove(2); E) liste.RemoveAt("Can");

16.

```
ArrayList liste = new ArrayList();
liste.add("Ali");
liste.add("Elif");
liste.add("Can");
liste.add("Fatih");
liste.add("Mehmet");

for (int index = 0; index < liste._____; index++)
{
   lbListe.Items.Add(liste[_____]);
}
```

Yukarıda yer alan kodlamalarda boş bırakılan yerleri sırasıyla aşağıdakilerden hangisi doğru şekilde tamamlar?

A) Length, index B) Count, index C) Size, liste D) Count, liste E) Size, index

17. Queue koleksiyonu için aşağıdaki ifadelerden hangisi doğrudur?

A) Her eklenen eleman koleksiyon tarafından otomatik olarak sıralanır.
B) İlk giren eleman son çıkar.
C) Pop() ve Push() metotları kullanılır.
D) Anahtar (Key)-Değer (Value) ikilisine sahiptir.
E) İlk giren eleman ilk çıkar.

18. Aşağıdakilerden hangisinde Dictionary nesnesinin tanımlaması doğru olarak yapılmıştır?

A) Dictionary[int,string] nesne = new Dictionary[int, string]();
B) Dictionary<int,string> nesne = new Dictionary(int, string);
C) Dictionary(int,string) nesne = new Dictionary<int, string>();
D) Dictionary<int,string> nesne = new Dictionary<int, string>();
E) Dictionary<int,string> nesne = new Dictionary<int, string>;

19. Hashtable koleksiyonu ile Dictionary koleksiyonu arasındaki fark aşağıdaki seçeneklerden hangisinde verilmiştir?

	Hashtable Koleksiyonu	Dictionary Koleksiyonu
A)	İlk giren eleman ilk çıkar.	Son giren eleman ilk çıkar.
B)	Sıralama yapar.	Sıralama yapmaz.
C)	Veri tipleri belirtilmez.	Veri tipleri belirtilir.
D)	Anahtar değeri sayı olmalıdır.	Anahtar değeri sayı olmayabilir.
E)	Aynı Key değeri verilebilir.	Aynı Key değeri verilemez.

5. ÖĞRENME BİRİMİ

FORM UYGULAMALARI

KONULAR

5.1. FORMLAR VE ÖZELLİKLERİ
5.2. İLETİŞİM KUTULARI

ANAHTAR KELİMELER

Dizi, index, veri tipi, for döngüsü, foreach döngüsü, koleksiyon

NELER ÖĞRENECEKSİNİZ?

- Form sınıfı ve kontrol sınıfı kavramları
- Form uygulamaları
- Masaüstü uygulamaları için menü oluşturma
- Veri doğrulama işlemleri
- Kontrollere veri bağlama işlemleri

HAZIRLIK ÇALIŞMALARI

1. Windows ortamında çalışan uygulamalardan hatırladıklarınızı arkadaşlarınızla paylaşınız.
2. Windows uygulamalarında bulunan nesneler neler olabilir? Araştırınız.

5.1. FORMLAR

Herhangi bir etkileşimli uygulamada en az bir tane kullanıcı arabirimi bulunur. C# programlama dili ile geliştirilen Windows form uygulamalarında bu arabirimler için Form sınıfından üretilen nesneler kullanılır. Formlar; TextBox, Button, Label gibi kontrolleri üzerinde tutarak bir arabirim oluşturmak için kullanılır. Kod editörü ile Windows form uygulama projesi hazırlandığında Form1 ismindeki form otomatik olarak oluşturulur.

Tüm uygulamalar, belirlenen bir noktadan itibaren çalışmaya başlar. Windows form uygulamalarında başlangıç noktası, Solution Explorer penceresinde Program.cs dosyası içindeki Main isimli fonksiyon ile belirlenir.

```
static void Main()
{
   Application.EnableVisualStyles();
   Application.SetCompatibleTextRenderingDefault(false);
   Application.Run(new Form1());
}
```

Program.cs dosyasındaki kodlamalarda **Application.Run** metodu sayesinde uygulamanın hangi form ile başlayacağı belirtilir.

Windows form uygulamalarında kullanılan formlar ve kontroller, **Form** sınıfından oluşturulan nesnelerdir. Bu nesnelerin ait oldukları sınıfın özellikleri (properties), olayları (events) ve metotları (methods) uygulama içinde ihtiyaca göre kullanılır. Form ile kontrollerin bazı özellik, olay ve metotları aynı, bazıları ise farklıdır.

5.1.1. Form Sınıfı

Windows form uygulamalarında System.Windows.Forms isim uzayındaki Form sınıfından üretilen form nesneleri kullanılır. Projeye bir form eklendiğinde Form sınıfının özeliklerini miras alan yeni Form sınıfı, kod editörü tarafından oluşturulur.

```
public partial class Form1 : Form
```

Oluşturulan form sınıfının içinde sınıf ismiyle aynı olan **Yapıcı Metot** (Contructor) bulunur. Yapıcı Metot içinden InitializeComponent isminde bir metot çağrılır. InitializeComponent metodu form ile birlikte oluşturulan Form_İsmi.Designer.cs dosyası içinde bulunur. Bu metot, form tasarımı sırasında form içine kontroller eklendiğinde veya kontrollerin özellikleri değiştirildiğinde arka planda kodlar üzerinde güncelleme yapar. Form açıldığında form ve forma eklenen kontrollerin özellikleri ve olayları bu metot

içindeki kodlara göre düzenlenir. Örneğin formun boyutları görsel olarak değiştirildiğinde, forma bir kontrol eklendiğinde veya özelliklere değer verildiğinde kod editörü tarafından InitializeComponent metodu otomatik olarak güncellenir. Tablo 5.1, Tablo 5.2 ve Tablo 5.3'te Form sınıfında kullanılan temel özellikler, metotlar ve olaylar verilmiştir.

Tablo 5.1: Form Sınıfının Temel Özellikleri

AcceptButton	Form aktif iken enter tuşuna basıldığında belirlenen bir butona tıklama işlemi yapar.
CancelButton	Form aktif iken esc tuşuna basıldığında belirlenen bir butona tıklama işlemi yapar.
ControlBox	Formun sağ üst köşesindeki kontrol butonlarını gösterir veya gizler.
FormBorderStyle	Formun kenarlık stilini ayarlar.
MinimizeBox	Formun sağ üst köşesindeki küçültme butonunu aktif veya pasif yapar.
MaximizeBox	Formun sağ üst köşesindeki büyültme butonunu aktif veya pasif yapar.
ShowInTaskbar	Görev çubuğunda formun görünüp görünmeyeceğini belirler.
StartPosition	Çalışma anında formun başlangıç konumunu belirler.
WindowState	Başlangıçta formun nasıl görüntüleneceğini ayarlar.

Tablo 5.2: Form Sınıfının Temel Metodları

Activate()	Form aktif hâle gelir.
Close()	Formu kapatır. Form, başlangıç formu ise uygulamayı sonlandırır.
CenterToScreen()	Formu ekranın merkezine yerleştirir.
Show()	Formu gösterir.
ShowDialog()	Formu diyalog kutusu olarak gösterir. Gösterilen form kapatılmadan diğer forma geçiş yapılamaz.
Hide()	Formu gizler.

Tablo 5.3: Form Sınıfının Temel Olayları

Load	Form açılırken gerçekleşir.
Click	Form üzerine tıklandığı zaman gerçekleşir.
FormClosing	Form kapanmadan hemen önce gerçekleşir.
FormClosed	Form kapandıktan sonra gerçekleşir.

1. Uygulama

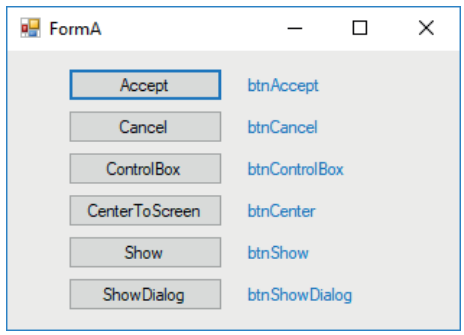
http://kitap.eba.gov.tr/KodSor.php?KOD=21081

Form Sınıfı

Bu uygulamada Form sınıfı için bazı olayların ve özelliklerin kullanılması gösterilecektir.

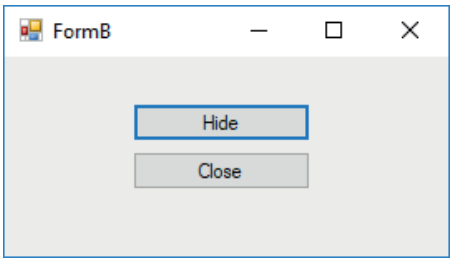
1. Adım: Yeni bir proje oluşturarak FormA ve FormB isminde iki form hazırlayınız.

2. Adım: FormA için Görsel 5.1'deki form tasarımını yapınız ve kontrollere name değerlerini veriniz.



Görsel 5.1: Form sınıfı uygulaması birinci form tasarımı

3. Adım: FormB için Görsel 5.2'deki form tasarımını yapınız ve kontrollere name değerlerini veriniz.



Görsel 5.2: Form sınıfı uygulaması ikinci form tasarımı

4. Adım: Birinci formdaki Accept ve Cancel butonları Click olaylarına aşağıdaki kodlamaları yapınız. Kodlamaları yaptıktan sonra Properties penceresinden formun AcceptButton ve CancelButton özelliğine ilgili butonları atayınız.

```
private void btnAccept_Click(object sender, EventArgs e)
 {
   MessageBox.Show("Enter tuşu çalışıyor.");
 }
 private void btnCancel_Click(object sender, EventArgs e)
 {
    MessageBox.Show("Esc tuşu çalışıyor.");
 }
```

5. Adım: ControlBox butonu Click olayına formun sağ üst köşesindeki butonların gösterilmesi veya gizlenmesi işlemini gerçekleştiren kodlamayı yapınız.

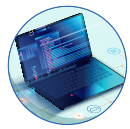
```
private void btnControlBox_Click(object sender, EventArgs e)
{
   if (this.ControlBox == true)
      this.ControlBox = false;
   else
      this.ControlBox = true;
}
```

6. Adım: CenterToScreen butonu Click olayına formu ekranın merkezine yerleştirme işlemini gerçekleştiren kodlamayı yapınız.

```
private void btnCenter_Click(object sender, EventArgs e)
{
   this.CenterToScreen();
}
```

7. Adım: Show butonu Click olayına oluşturulan ikinci formun gösterilmesini sağlayan kodlamayı yapınız.

```
private void btnShow_Click(object sender, EventArgs e)
{
   FormB formB = new FormB();
   formB.Show();
}
```



8. Adım: ShowDialog butonu Click olayına oluşturulan ikinci formun diyalog penceresi olarak gösterilmesini sağlayan kodlamayı yapınız.

```
private void btnShowDialog_Click(object sender, EventArgs e)
{
  FormB formB = new FormB();
  formB.ShowDialog();
}
```

9. Adım: İkinci formun Load olayına form yüklendiğinde çalışan kodlamaları yapınız.

```
private void FormB_Load(object sender, EventArgs e)
{
   MessageBox.Show("FormB yükleniyor.");
}
```

10. Adım: İkinci formun FormClosing olayına form kapanırken çalışan kodlamaları yapınız.

```
private void FormB_FormClosing(object sender, FormClosingEventArgs e)
{
  MessageBox.Show("FormB kapanıyor.");
}
```

11. Adım: İkinci formun FormClosed olayına form kapandıktan sonra çalışan kodlamaları yapınız.

```
private void FormB_FormClosed(object sender, FormClosedEventArgs e)
{
  MessageBox.Show("FormB kapandı.");
}
```

12. Adım: İkinci form içindeki Hide butonu Click olayına formun gizlenmesini sağlayan kodlamaları yapınız.

```
private void btnHide_Click(object sender, EventArgs e)
{
  this.Hide();
}
```

13. Adım: İkinci form içindeki Close butonu Click olayına formun kapanmasını sağlayan kodlamaları yapınız.

```
private void btnClose_Click(object sender, EventArgs e)
{
  this.Close();
}
```

14. Adım: İkinci formun Click olayına form üzerinde herhangi bir yere fare ile tıklandığında çalışan kodlamaları yapınız.

```
private void FormB_Click(object sender, EventArgs e)
{
   MessageBox.Show("FormB click olayı çalışıyor.");
}
```

5.1.2. Kontrol Sınıfı

Kontrol sınıfı, System.Windows.Forms isim uzayında yer alan bir sınıftır. Form içine eklenen Button, TextBox, ListBox gibi görsel Windows kontrollerinin temel sınıfıdır. Tablo 5.4 ve Tablo 5.5'te kontrol sınıfında kullanılan temel özellikler ve olaylar verilmiştir.

Tablo 5.4: Kontrollerin Temel Özellikleri

AcceptButton	Form aktif iken enter tuşuna basıldığında belirlenen bir butona tıklama işlemi yapar.
CancelButton	Form aktif iken esc tuşuna basıldığında belirlenen bir butona tıklama işlemi yapar.
ControlBox	Formun sağ üst köşesindeki kontrol butonlarını gösterir veya gizler.
FormBorderStyle	Formun kenarlık stilini ayarlar.
MinimizeBox	Formun sağ üst köşesindeki küçültme butonunu aktif veya pasif yapar.
MaximizeBox	Formun sağ üst köşesindeki büyültme butonunu aktif veya pasif yapar.
ShowInTaskbar	Görev çubuğunda formun görünüp görünmeyeceğini belirler.
StartPosition	Çalışma anında formun başlangıç konumunu belirler.
WindowState	Başlangıçta formun nasıl görüntüleneceğini ayarlar.

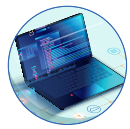
Tablo 5.5: Kontrollerin Temel Olayları

Click **DoubleClick** **MouseEnter** **MouseLeave** **MouseDown** **MouseUp** **MouseMove** **MouseHover** **MouseWheel**	Fare ile etkileşimi gerçekleştirir.
KeyPress **KeyUp** **KeyDown**	Klavye ile etkileşimi gerçekleştirir.
DragDrop **DragEnter** **DragLeave** **DragOver**	Sürükle bırak olaylarını gerçekleştirir.

2. Uygulama

Form Uygulamaları

Bazen form özelliklerine değer verilmesinde veya kontrollerin oluşturulmasında görsel kullanılarak değil, çalışma anında dinamik olarak kodlarla oluşturma ihtiyacı gerekebilir. Bu uygulamada birer adet TextBox, Label ve Button kontrolünü form içine görsel olarak Toolbox penceresinde eklemek yerine çalışma anında dinamik olarak ekleme işlemi yapılacaktır. Dinamik olarak eklenen kontrole özellik değerleri verme ve olay metotları oluşturma gibi işlemler gerçekleştirilecektir.



1. Adım: Yeni bir form uygulama projesi oluşturunuz. Projedeki formu kod görünümünde açınız.

2. Adım: Form içine eklenecek kontrolleri tanımlamak için Form sınıfının başlangıcına aşağıdaki kodlamaları yapınız.

```
private void FormB_Click(object sender, EventArgs e)
{
    MessageBox.Show("FormB click olayı çalışıyor.");
}
```

3. Adım: Form Load olayına form özelliklerini değiştirmek için gereken kodlamaları yazınız.

```
private void Form1_Load(object sender, EventArgs e)
{
    this.Text = "Form Sınıfı Uygulama 2";
    this.BackColor = Color.Green;
    this.ForeColor = Color.Black;
    this.Size = new Size(300, 150);
}
```

4. Adım: Forma eklenecek Label kontrolünün özellikleri için Form Load olayına kodları yazınız.

```
lblAdSoyad.Text = "Adınız";
lblAdSoyad.Location = new Point(10, 10);
lblAdSoyad.Size = new Size(65, 15);
lblAdSoyad.ForeColor = Color.White;
```

5. Adım: Forma eklenecek TextBox kontrolünün özellikleri için Form Load olayına kodları yazınız.

```
txtAdSoyad.Location = new Point(75, 10);
txtAdSoyad.Size = new Size(150, 15);
txtAdSoyad.TabIndex = 1;
```

6. Adım: Forma eklenecek Button kontrolünün özellikleri için Form Load olayına kodları yazınız.

```
btnGiris.Text = "Tıkla";
btnGiris.Location = new Point(100, 60);
btnGiris.Size = new Size(100, 30);
btnGiris.ForeColor = Color.White;
btnGiris.BackColor = Color.Black;
btnGiris.TabIndex = 2;
```

7. Adım: Buton için tıklama olayı ve TextBox için klavyeden bir tuşa basma olayı metotları oluşturulacaktır. Bu işlem için kontrole atanan olaydan sonra += operatörü ile olay metotlarının isimlerini vererek Form Load olayı içine ekleyiniz (+= operatöründen sonra tab tuşuna basıldığında metot otomatik olarak oluşturulacaktır.).

```
txtAdSoyad.KeyPress += TxtAdSoyad_KeyPress;
btnGiris.Click += BtnGiris_Click;
```

8. Adım: Dinamik oluşturulan, özellikleri verilen ve olay metotları ataması yapılan kontrollerin forma eklenmesi için Form Load olayına kodlamaları yapınız.

```
this.Controls.Add(lblAdSoyad);
this.Controls.Add(txtAdSoyad);
this.Controls.Add(btnGiris);
```

9. Adım: TextBox için oluşturulan Keypress olayında klavyeden girilen her harfi büyük harfe çeviren kodlamaları yapınız.

```
private void TxtAdSoyad_KeyPress(object sender, KeyPressEventArgs e)
{
        e.KeyChar = Char.ToUpper(e.KeyChar);
}
```

10. Adım: Buton için oluşturulan Click olayında TextBox içine girilen bilgileri mesaj kutusunda gösteren kodlamaları yapınız.

```
private void BtnGiris_Click(object sender, EventArgs e)
{
  MessageBox.Show("Merhaba " + txtAdSoyad.Text);
}
```

Sıra Sizde

Kodlamalarda neden this ifadesi kullanılmıştır? Açıklayınız.

5.1.3. Konteyner Kontrolleri

Bu kontroller, diğer kontroller için özelleştirilebilir konteyner görevi gören özel kontrollerdir. Form uygulaması geliştirilirken eklenen kontrollerin fazlalığı, formun görüntüsünü karmaşık hâle getirebilir. Formun karmaşıklığını önlemek ve birbirleriyle ilişkili kontrolleri bir arada tutmak için konteyner kontroller kullanılır. Bunlar; Toolbox penceresinde Containers sekmesi içinde bulunan Panel, Groupbox, TabControl, FlowLayoutPanel kontrolleridir. Örneğin öğrenci ve veli bilgileri kaydedilen bir arabirim formu tasarlandığında öğrenci bilgilerine ait kontroller bir konteyner içinde, veli bilgilerine ait kontroller ise başka bir konteyner içinde tutulabilir.

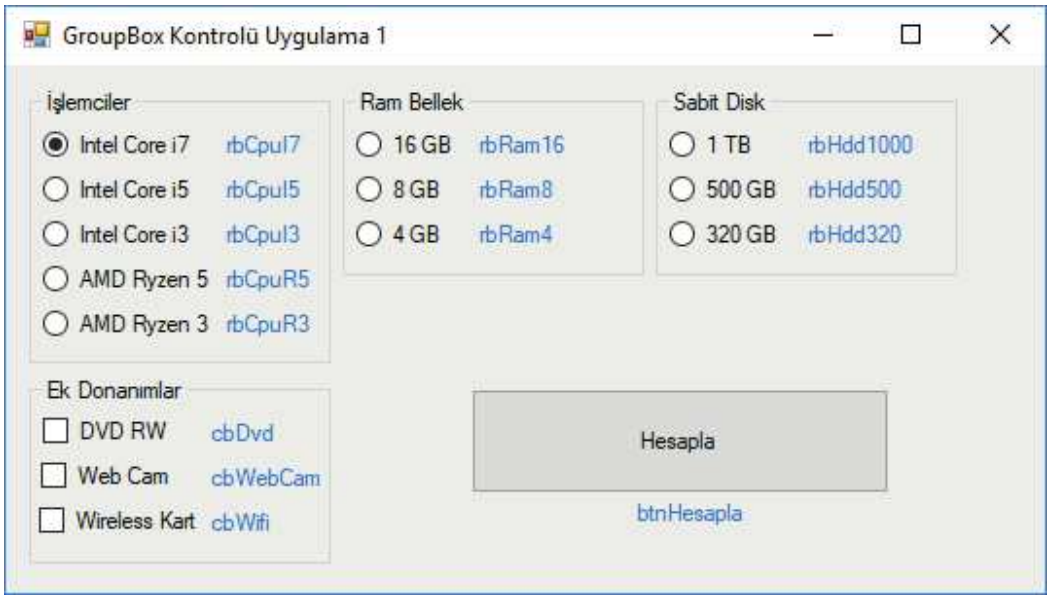
Konteyner kontrolünün belirli özellikleri değiştirildiğinde içinde bulunan kontrollerin özellikleri de bu durumdan etkilenir. Konteyner kontrolün **Enable** veya **Visible** özellikleri değiştirildiğinde konteyner içindeki tüm kontroller aynı şekilde etkilenir.

GroupBox Kontrolü: Formu bölümlere ayırarak ilişkili kontrolleri bir arada tutmak için kullanılan bir konteyner kontrolüdür.

3. Uygulama

Bu uygulamada form üzerindeki ilişkili kontrolleri GroupBox içine alarak basit bir bilgisayar fiyat hesaplama uygulaması yapılacaktır.

1. Adım: Görsel 5.3'teki form tasarımını yapınız ve kontrollere name değerlerini veriniz.



Görsel 5.3: GroupBox uygulaması form tasarımı

2. Adım: Hesapla butonu Click olayına belirlenen taban fiyata, seçilen her donanımın fiyatını ekleyen ve toplam fiyatı hesaplayıp, mesaj olarak gösteren kodlamayı yapınız.

```
private void btnHesapla_Click(object sender, EventArgs e)
{
  decimal tabanFiyat = 500;
  // İşlemci fiyat hesaplaması
  decimal cpuFiyat = 0;
  if (rbCpuI7.Checked)
    cpuFiyat = 300;
  else if (rbCpuI5.Checked)
    cpuFiyat = 200;
  else if (rbCpuI3.Checked)
    cpuFiyat = 100;
  else if (rbCpuR5.Checked)
     cpuFiyat = 250;
  else if (rbCpuR3.Checked)
     cpuFiyat = 150;
  tabanFiyat += cpuFiyat;
  // Ram bellek fiyat hesaplaması
  decimal ramFiyat = 0;
  if (rbRam16.Checked)
    ramFiyat = 125;
  else if (rbRam8.Checked)
    ramFiyat = 75;
  else if (rbRam4.Checked)
     ramFiyat = 45;
  tabanFiyat += ramFiyat;
  MessageBox.Show(string.Format("Toplam Fiyat ={0:C}", tabanFiyat));
}
```

Sıra Sizde

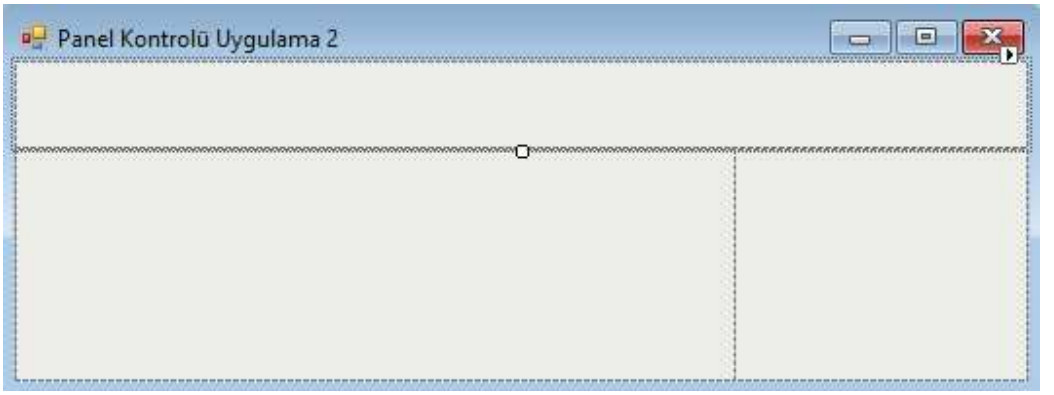
Üçüncü uygulamadaki diğer donanım özellikleri için fiyat hesaplamasını yapınız.

Panel Kontrolü: Bu kontrol, GroupBox gibi diğer kontrolleri barındıran formun alt bölümüdür. GroupBox'tan farklı olarak bu kontrolde kaydırma çubukları bulunur. **AutoScroll** özelliği **true** yapıldığında sınırları dışındaki alanlar için yatay veya dikey kaydırma çubukları görünür hâle gelir.

4. Uygulama

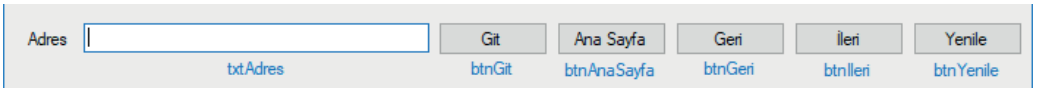
Bu uygulamada kod editörü içindeki Toolbox penceresinde bulunan WebBrowser ve Panel kontrolleri ekleyenerek Internet Explorer'a ait özellikleri kullanan web tarayıcı uygulaması yapılacaktır.

1. Adım: Yeni bir Windows form uygulaması oluşturunuz. Projenizin formuna üç adet Panel kontrolü ekleyiniz. Panellerin Dock özelliğini Görsel 5.4'teki gibi Top, Fill ve Right olarak düzenleyiniz.



Görsel 5.4: Panel kontrollerinin form üzerinde yerleşimi

2. Adım: Üst panel için Görsel 5.5'teki tasarımı yapınız.



Görsel 5.5: Web tarayıcı gezinti bölümü panel tasarımı

3. Adım: Toolbox penceresinden WebBrowser kontrolünü ortadaki panel içine sürükleyerek bırakınız. Properties penceresinden ScriptErrorsSuppressed özelliğini true yapınız.

4. Adım: Toolbox penceresinden ListBox kontrolünü sağdaki panel içine sürükleyerek bırakınız. ListBox kontrolünün Dock özelliğini Fill yapınız.

5. Adım: Üst panelde bulunan Git ve Ana Sayfa butonlarının Click olaylarına aşağıdaki kodlamaları yapınız.

```
private void btnGit_Click(object sender, EventArgs e)
{
webBrowser1.Navigate(txtAdres.Text);
}
private void btnAnaSayfa_Click(object sender, EventArgs e)
{
webBrowser1.GoHome();
}
```

6. Adım: Gezilen sayfaların listesini ListBox kontrolü içinde göstermek için WebBrowser kontrolünün Navigated olayına aşağıdaki kodlamaları yapınız.

```
private void webBrowser1_Navigated(object sender, WebBrowserNavigatedEventArgs e)
{
  listBox1.Items.Add(webBrowser1.Url);
}
```

Sıra Sizde

1. Geri, İleri ve Yenile butonlarına tıklandığında WebBrowser kontrolünün ziyaret edilen siteler arasında gezinti yapmasını ve mevcut siteyi yenilemesini sağlayan kodlamaları yapınız.
2. Üstteki panel içine Geçmiş adında bir buton ekleyerek ListBox kontrolünün bulunduğu paneli gösterip gizleme işlemini gerçekleştiren kodlamaları yapınız.

TabControl Kontrolü: Bu kontrol, gruplar hâlindeki kontrolleri sekme sayfalarında göstermek için kullanılır. TabControl sekmelerine tıklanarak sekme sayfaları arasında geçiş yapılabilir.

5. Uygulama

Bu uygulamada TabControl kullanılarak müşteri bilgileri girilecek ve verilen siparişlere göre hesap oluşturulacaktır.

1. Adım: Yeni bir form uygulama projesi oluşturunuz. Projenizin formuna TabControl ekleyiniz.

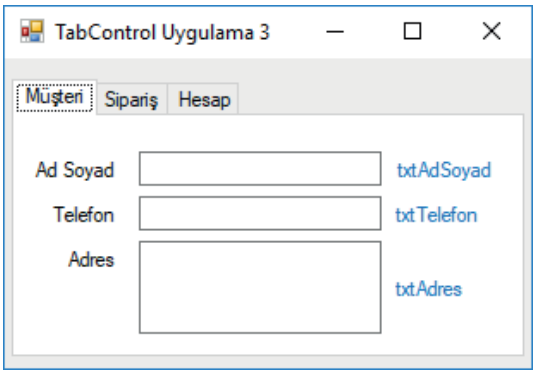
2. Adım: TabControl nesnesini seçtikten sonra Properties penceresinden TabPages özelliğine tıklayarak açılan pencereden üç adet sekme sayfası oluşturunuz. Oluşturulan sekme sayfalarının Text özelliğine sırasıyla Müşteri, Sipariş ve Hesap değerlerini atayınız.

3. Adım: İlk sekme sayfası için Görsel 5.6'daki tasarımı yapınız.

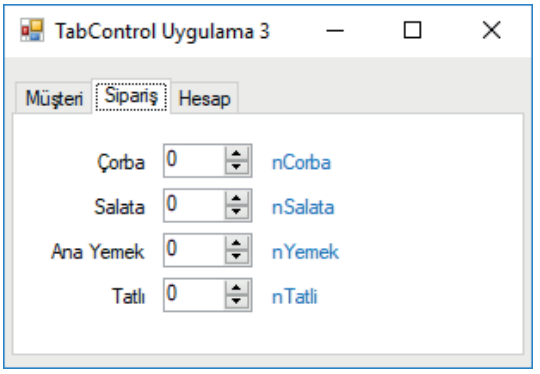
4. Adım: İkinci sekme sayfası için Görsel 5.7'deki tasarımı yapınız.

5. Adım: Üçüncü sekme sayfası için txtBilgi name değerine sahip bir TextBox ekleyerek Dock özelliğini Fill olarak değiştiriniz.

6. Adım: TabControl nesnesinin SelectedIndexChanged olayına üçüncü sekme sayfası açıldığında diğer sekme sayfalarındaki bilgileri kullanarak hesaplayan kodlamaları yapınız.



Görsel 5.6: TabControl sekme sayfası tasarımı



Görsel 5.7: TabControl sekme sayfası tasarımı

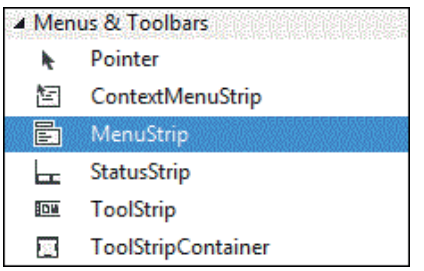
```
private void tabControl1_SelectedIndexChanged(object sender, EventArgs e)
{
   if(tabControl1.SelectedIndex==2)
   {
      txtBilgi.Text = "";
      txtBilgi.Text += txtAdSoyad.Text + "\r\n";
      txtBilgi.Text += txtTelefon.Text + "\r\n";
      txtBilgi.Text += txtAdres.Text + "\r\n";
      decimal hesap = 0;
      if (nCorba.Value > 0)
      {
         hesap += nCorba.Value*12;//Adet fiyatı 12 lira.
         txtBilgi.Text += string.Format("Çorba {0:C}",nCorba.Value * 12) + "\r\n";
      }
      txtBilgi.Text += "----------------";
      txtBilgi.Text += string.Format("Toplam {0:C}",hesap);
   }
}
```

Sıra Sizde

Müşterilerin diğer siparişleri için hesaplama kodlamalarını yapınız.

5.2. MENÜLER

Menüler, bilgisayar uygulamalarında benzer veya ilgili komutları gruplandırmak için kullanılır. Menüler, uygulamanın daha kolay kullanılmasına olanak sağlar. Bu bölümde form içine menü ekleme, menü elemanlarına isim verme, menü elemanlarına tıklandığında ilgili kodları çalıştırma gibi işlemler açıklanacaktır. Bir Windows form uygulamasına menü eklemek için Görsel 5.8'de gösterilen Toolbox penceresinde Menus & Toolbars sekmesindeki menü kontrolleri kullanılır.



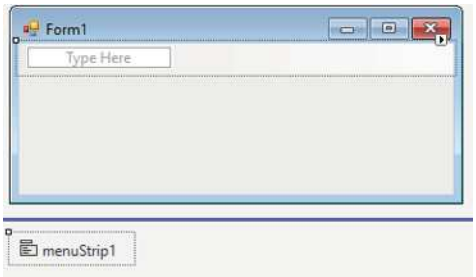
Görsel 5.8: Menüler ve araç çubukları

5.2.1. MenuStrip Kontrolü

MenuStrip kontrolü, menü araçlarının içinde en çok kullanılan kontroldür. Bu kontrol; birçok uygulamada karşılaşılan, genelde uygulamanın üst kısmında yer alan menüler ve alt menülerden oluşur. MenuStrip kontrolü, form tasarım görünümünde iken Toolbox penceresinden çift tıklanarak ya da form üzerine sürüklenerek forma eklenebilir.

MenuStrip kontrolü form içine eklendiğinde menülerin tasarlanacağı alan formun üst kısmında belirir. Bu alanda menüler görsel olarak tasarlanabilir. Menüler, Properties içinde Items özelliğine tıklanarak açılan "Items Collection Editor" penceresinden de tasarlanabilir.

Menüleri tasarım görünümünde oluşturmak için Görsel 5.9'da gösterilen "Type Here" yazan kısımlara bir kez tıklandığında menü içeriklerinin oluşturulmasını sağlayan kutular belirir. Bu kutulardan menü elemanı, açılır kutu, ayraç veya yazı eklenebilir.



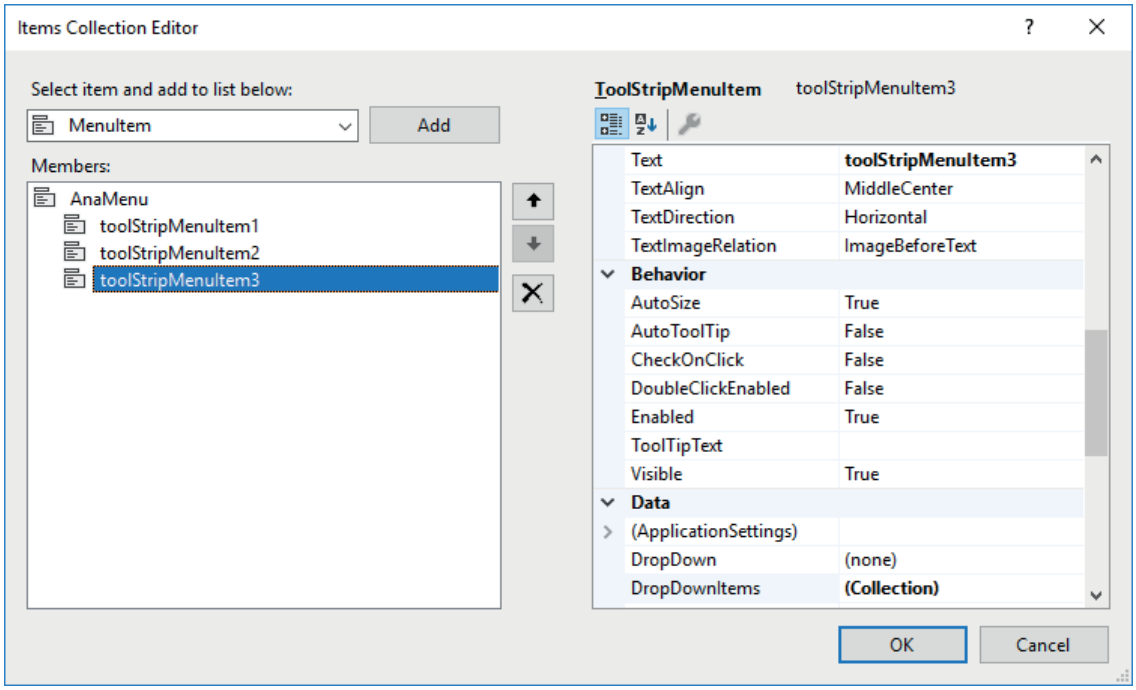
Görsel 5.9: MenuStrip tasarım alanı ve nesnesi

6. Uygulama

Bu uygulamada form içine MenuStrip kontrolü eklenerek menü elemanlarını oluşturma işlemleri yapılacaktır.

1. Adım: Form üzerine Toolbox içinden MenuStrip kontrolü ekleyiniz. MenuStrip kontrolünün name değerini "AnaMenu" olarak değiştiriniz.

2. Adım: MenuStrip kontrolü Items özelliğine tıklayarak açılan "Items Collection Editor" penceresinden Görsel 5.10'da olduğu gibi üç tane MenuItem ekleyiniz. Eklenen menü elemanlarının Text özelliklerini sırasıyla Dosya, Düzen, Çıkış ve Name özelliklerini de menuDosya, menuDuzen, menuCikis olarak değiştiriniz.



Görsel 5.10: Items Collection Editor penceresi

Not

"Items Collection Editor" kullanılarak hazırlanan menü tasarımı, menü alanı içindeki "Type Here" kutuları seçilerek de yapılabilir.

Menü elemanlarına açılır menü ekleyerek benzer görevleri yerine getiren alt menüler oluşturulabilir. Örneğin Dosya menüsünde dosya açma, kaydetme, yazıcıya gönderme gibi görevleri yerine getirecek işlemler için açılır menü kullanılır.

Menü elemanlarının altında açılır menü oluşturma, görsel olarak menü elemanlarının altında beliren kutularla yapılabileceği gibi menü elemanı seçildikten sonra Properties penceresi içinden DropDownItems özelliği kullanılarak da yapılabilir.

7. Uygulama

Açılır Menüler

Bu uygulamada menü elemanlarına tıklandığında benzer görevleri yerine getirecek olan açılır menü oluşturulacaktır. Oluşturulan açılır menülerin Text ve Name değerleri Görsel 5.11'de verilmiştir.

Dosya (menuDosya)	Düzen (menuDuzen)	Çıkış (menuCikis)
Yeni (menuYeni)	Kes (menuKes)	
Aç (menuAc)	Kopyala (menuKopya)	
Kaydet (menuKaydet)	Yapıştır (menuYapistir)	
Yazdır (menuYazdir)		

Görsel 5.11: Açılır menüler

1. Adım: Dosya menü elemanı için açılır menüleri görsel tasarım alanındaki kutuları kullanarak oluşturunuz.

2. Adım: Düzen menü elemanı için açılır menüleri Properties içinde DropDownItems özelliğini kullanarak oluşturunuz.

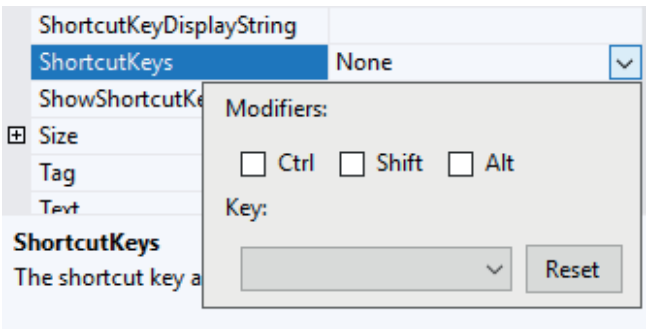
Menü Elemanlara Kısayol Tuşlarının Atanması

Kısayol tuşları, menü elemanlarına tıklanmadan klavye üzerindeki tuş birleşimleri ile menü elemanlarına erişim sağlar. Yaygın olarak kullanılan tuş birleşimlerinden bazıları Tablo 5.6'da verilmiştir.

Tablo 5.6: Yaygın Kullanılan Tuş Birleşimleri

Ctrl+N	Yeni Dosya	Ctrl+C	Kopyala
Ctrl+O	Dosya Açma	Ctrl+X	Kes
Ctrl+S	Kaydetme	Ctrl+V	Yapıştır
Ctrl+P	Yazdır	Alt+F4	Programı Kapat

Menü elemanlarına Properties penceresinden ShortcutKeys özelliğiyle Görsel 5.12'de olduğu gibi kısayol ataması gerçekleştirilebilir.



Görsel 5.12: Menülere kısayol ataması

Pasif Menü Elemanları

Bir menü elemanı soluk renkte pasif hâle Properties penceresinden Enable özelliği false yapılarak getirilir. Örneğin bir metin editörü uygulamasında yapıştırma işlemi, kopyalama veya kesme işlemleri yapılıncaya kadar pasif hâldedir. Kopyalama veya kesme işlemi yapılırsa Yapıştır menü elemanı aktif hâle gelir.

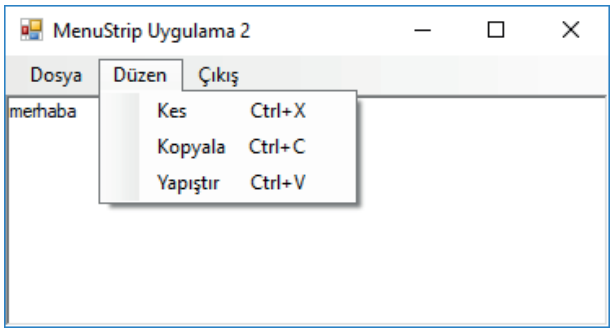
8. Uygulama

Editör Uygulaması

Yedinci uygulamadaki menü tasarımı kullanılarak form içine eklenen RichTextBox kontrolü ile metinleri kesme, kopyalama ve yapıştırma işlemlerini gerçekleştiren bir editör uygulaması yapılacaktır.

1. Adım: Form içine Toolbox penceresinde RichTextBox kontrolü ekleyiniz. Görsel 5.13'teki gibi RichTextBox kontrolünün formun tamamını kaplaması için Properties penceresinde Dock özelliğini Fill olarak değiştiriniz.

2. Adım: Eklenen RichTextBox kontrolünün name değerini txtEditor olarak değiştiriniz. RichTextBox içinden seçilen metnin kesme, kopyalama ve yapıştırma işlemleri için menü elemanları Click olaylarına kodlamaları yapınız.



Görsel 5.13: Menülere kısayol ataması

```
private void menuKes_Click(object sender, EventArgs e)
{
  txtEditor.Cut();
}
private void menuKopyala_Click(object sender, EventArgs e)
{
  txtEditor.Copy();
}
private void menuYapistir_Click(object sender, EventArgs e)
{
  txtEditor.Paste();
}
```

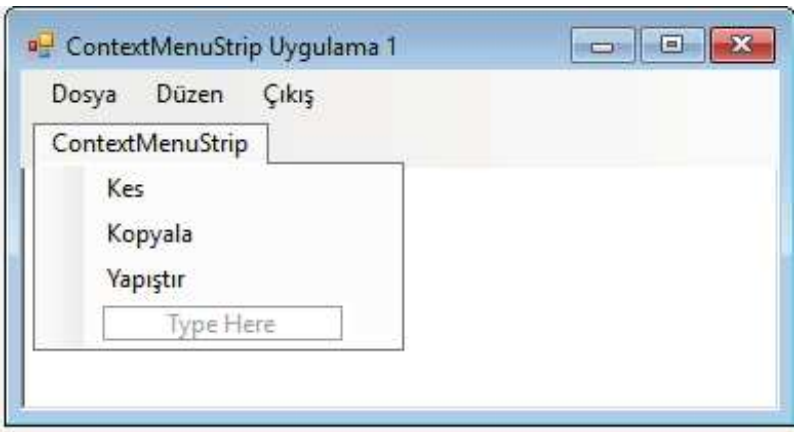
5.2.2. ContextMenuStrip Kontrolü

Birçok uygulamada işlemlerin daha hızlı gerçekleştirilebilmesi için sağ tıklama ile açılır menüler kullanılır. Bu menüleri oluşturmak için ContextMenuStrip kontrolü kullanılır. ContextMenuStrip tasarımı ve menü elemanlarının oluşturulması, MenuStrip kontrolüne benzer. Toolbox içinden Menus & Toolbars sekmesindeki ContextMenuStrip kontrolü çift tıklanarak veya sürüklenerek form içine eklenir. Menüler oluşturulduktan sonra bu menüler, formla veya form içindeki herhangi bir kontrolle ilişkilendirilir. Uygulama çalıştırıldığında ilişkilendirilen kontrol üzerindeyken farenin sağ tuşuna basıldığında menü görünür hâle gelir.

9. Uygulama

Editör uygulamasında oluşturulan form üzerine ContextMenuStrip kontrolü eklenerek seçilen metin üzerinde sağ tıklandığında açılır menüleri gösterme işlemi yapılacaktır. Açılır menüden kesme, kopyalama ve yapıştırma işlemleri gerçekleştirilecektir.

1. Adım: Form içine Toolbox penceresinden Menus & Toolbars sekmesindeki ContextMenuStrip kontrolünü ekleyiniz. ContextMenuStrip kontrolünün name değerini cmDuzen olarak değiştiriniz. Açılır menü tasarımını Görsel 5.14'te olduğu gibi yapınız.



Görsel 5.14: ContextMenuStrip kontrolü menü tasarımı

2. Adım: ContextMenuStrip içindeki menü elemanları Click olayına sekizinci uygulamadaki düzen menüsü için yapılan olayları seçiniz (Burada seçim işleminin yapılması kod tekrarını önlemek içindir.).

3. Adım: ContextMenuStrip menülerini RichTextBox nesnesi ile ilişkilendirmek için RichTextBox nesnesini seçtikten sonra Properties penceresinden ContextMenuStrip özelliğini tıklayarak açılır menüyü seçiniz.

5.3. İLETİŞİM KUTULARI (DIALOG BOXES)

Dialog Boxes, kullanıcı ile uygulama arasında iletişimi sağlayan pencerelerdir. İletişim kutuları genellikle kullanıcıya bir komutun nasıl işleneceğini ifade eder veya kullanıcıdan bir sorunun cevabı istenildiğinde kullanılır.

İletişim kutuları yeniden boyutlandırılamaz. İletişim kutuları; kullanıcıya bilgi, hata veya uyarı mesajı göndermede, dosya açma veya kaydetme esnasında dosya konumunu belirlemede, belgeyi yazıcıya gönderme esnasında yazdırma ayarlarında, editör uygulamalarında yazının fontunu veya rengini değiştirmede vb. açılan pencerelerdir.

En çok kullanılan iletişim kutuları şunlardır:

- MessageBox
- OpenDialog
- SaveDialog
- FontDialog
- ColorDialog
- PrintDialog

5.3.1. Mesaj İletişim Kutusu (MessageBox)

MessageBox kullanıcıya bir mesaj vermek amacıyla kullanılır. Show() metodu ile parantez içine yazılan mesajları gösterir. Ayrıca overload yöntemi ile farklı kullanım şekilleri de vardır. Aşağıda MessageBox'ın temel kullanımı verilmiştir.

```
MessageBox.Show("Merhaba Dünya","Bilgi Mesajı");
```

İletişim kutusuna başlık eklemek için mesaj bilgisi verildikten sonra virgül bırakılarak başlık bilgisi yazılır.

```
MessageBox.Show("Merhaba Dünya");
```

İletişim kutusunda standart olarak Tamam butonu bulunur. Tamam butonu dışında buton eklemek istenildiğinde MessageBox butonları kullanılır. MessageBox butonları şunlardır:

- MessageBoxButtons.OK - Tamam
- MessageBoxButtons.AbortRetryIgnore - Durdur, Yeniden Dene, Yoksay
- MessageBoxButtons.OKCancel - Tamam, İptal
- MessageBoxButtons.RetryCancel - Yeniden Dene, İptal
- MessageBoxButtons.YesNo - Evet, Hayır
- MessageBoxButtons.YesNoCancel - Evet, Hayır, İptal

```
MessageBox.Show("Merhaba Dünya","Bilgi Mesajı",MessageBoxButtons.YesNoCancel);
```

İletişim kutusu içinde ikonlar kullanılabilir. Bu ikonlar, verilmek istenen mesajın içeriğiyle ilişkili olmalıdır. MessageBox ikonları şunlardır:

- **MessageBoxIcon.Error** – Hata ikonu
- **MessageBoxIcon.Warning** – Uyarı ikonu
- **MessageBoxIcon. Information** – Bilgilendirme ikonu
- **MessageBoxIcon.Question** – Soru ikonu

```
MessageBox.Show("Merhaba Dünya","Bilgi Mesajı",MessageBoxButtons.YesNoCancel,
MessageBoxIcon.Information);
```

Uygulama içinden kullanıcıya iletişim kutusu gösterildikten sonra kullanıcının hangi butona tıkladığının kontrolü için DialogResult kullanılır.

```
DialogResult cevap=MessageBox.Show("Bu dosyayı silmek istediğinize emin misiniz?", "Dosya Sil",
MessageBoxButtons.YesNo,MessageBoxIcon.Warning);
if(cevap==DialogResult.Yes)
{
   MessageBox.Show("Dosya silindi.");
}
else if(cevap==DialogResult.No)
{
   MessageBox.Show("İşlem iptal edildi.");
}
```

Sıra Sizde

Bir önceki bölümde yaptığınız editör uygulamasında Çıkış menü elemanına tıklandığında MessageBox iletişim kutusu gösterilerek "Çıkmak istediğinize emin misiniz?" mesajı verilecektir. Evet butonuna tıklandığında uygulamayı kapatma işlemini gerçekleştiren kodlamayı yapınız.

5.3.2. Dosya Kaydet İletişim Kutusu (SaveFileDialog)

SaveFileDialog, kullanıcının bilgisayarındaki klasörlere göz atmasını ve dosyayı kaydedebileceği konumu belirlemesini sağlayan iletişim kutusudur. SaveFileDialog dosya kaydetme işlemi yapmaz, sadece kaydedilecek dosyanın konumunun ve isminin belirlenmesini sağlar. Dosya iletişim kutusu, tasarım veya çalışma anında oluşturularak kullanılabilir. Tasarım ekranında iken Toolbox penceresi Dialogs sekmesinden SaveFileDialog form içine sürüklenerek veya çift tıklanarak eklenebilir. Çalışma anında ise SaveFileDialog sınıfından bir nesne oluşturularak iletişim kutusunun özellikleri kontrol edilebilir.

```
SaveFileDialog sfd = new SaveFileDialog();
sfd.ShowDialog();
```

Filter özelliği, SaveFileDialog sınıfından oluşturulan nesne veya Toolbox penceresinde form içine eklenen diyalog, iletişim kutusu kaydedilecek dosyanın uzantısını veya uzantılarını belirlemek için kullanılır.

```
sfd.Filter ="Text Dosyası |*.txt";
// Kaydedilecek dosyanın uzantısı txt olarak belirlenir.
sfd.Filter = "Text Dosyası |*.txt| Word Dosyası |*.docx";
// Kaydedilecek dosyanın uzantısı txt  veya docx olarak belirlenir.
```

SaveFileDialog penceresinde kaydedilecek dosyanın ismi, uzantısı ve konumu belirlendikten sonra Kaydet butonuna tıklandığını kontrol etmek için DialogResult kullanılır.

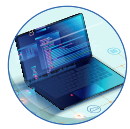
```
SaveFileDialog sfd = new SaveFileDialog();
sfd.Filter = "Text Dosyası |*.txt| Tüm Dosyalar |*.*";
DialogResult cevap = sfd.ShowDialog();
if(cevap==DialogResult.OK)
{
 MessageBox.Show("Dosya kaydetme işlemi başlıyor.")
}
```

10. Uygulama

SaveFileDialog

Burada Editör uygulamasındaki RichTextBox kontrolünün içinde yer alan metinleri bir metin dosyası olarak kaydetme işlemi gerçekleştirilecektir.

1. Adım: Editör uygulamasında Dosya menüsündeki Kaydet menü elemanı Click olayına SaveFileDialog sınıfından oluşturulan nesne kullanılarak iletişim kutusu gösterilecektir. İletişim kutusundaki Kaydet butonuna tıklandığında RichTextBox içindeki bilgileri belirlenen konuma SaveFile() metodu ile kaydetme işlemini gerçekleştiren kodlamaları yapınız.



```
private void menuKaydet_Click(object sender, EventArgs e)
{
  SaveFileDialog sfd = new SaveFileDialog();
  sfd.Filter = "Text Dosyası |*.txt| Tüm Dosyalar |*.*";
  DialogResult cevap = sfd.ShowDialog();
  if(cevap==DialogResult.OK)
  {
    txtEditor.SaveFile(sfd.FileName,RichTextBoxStreamType.PlainText);
  }
}
```

5.3.3. Dosya Aç İletişim Kutusu (OpenFileDialog)

OpenFileDialog, kullanıcıların kendi bilgisayarına veya ağdaki herhangi bir bilgisayarın klasörlerine göz atmasına ve açmak için bir dosya seçmesine olanak sağlayan iletişim kutusudur. OpenFileDialog sınıfından üretilen nesneyi veya Toolbox penceresi Dialogs sekmesinden eklenen kontrolü kullanarak seçilen dosyanın adı ve yolu elde edilir.

```
OpenFileDialog ofd = new OpenFileDialog();
ofd.ShowDialog();
```

OpenFileDialog, Filter özelliği ile iletişim kutusu içinde gösterilecek dosyaların uzantılarına göre filtreleme yaparak sadece belirlenen uzantıda olan dosyaların gösterilmesini sağlar.

```
ofd.Filter = "Text Dosyaları|*.txt";
// Uzantısı txt olanları gösterir.
ofd.Filter = "Text Dosyaları|*.txt| Word Dosyaları|*.docx";
// Uzantısı txt veya docx olanları gösterir.
ofd.Filter = "Text Dosyaları |*.txt| Word Dosyaları |*.docx| Tüm Dosyalar |*.*";
// Uzantısı txt, docx veya tüm dosyaları gösterir.
```

OpenFileDialog penceresinden dosya seçildikten sonra Dosya Aç butonuna tıklandığını kontrol etmek için DialogResult kullanılır.

```
DialogResult cevap= ofd.ShowDialog();
if(cevap==DialogResult.OK)
{
  MessageBox.Show(ofd.FileName);
}
```

11. Uygulama

OpenFileDialog

Burada Editör uygulaması kullanılarak bir metin dosyasının içeriğini RichTextBox kontrolü içinde gösterme işlemi yapılacaktır.

1. Adım: Editör uygulamasında Dosya menüsündeki Dosya Aç menü elemanı Click olayına OpenFileDialog nesnesi kullanılarak txt uzantılı dosyaların seçilmesi sağlanacaktır. İletişim kutusundaki Dosya Aç butonuna tıklandığında seçilen dosyanın RichTextBox kontrolünün LoadFile() metodu ile dosya içeriğini gösteren kodlamaları yapınız.

```
private void menuAc_Click(object sender, EventArgs e)
{
   OpenFileDialog ofd = new OpenFileDialog();
   ofd.Filter = "Text Dosyaları |*.txt";
   DialogResult cevap= ofd.ShowDialog();
   if(cevap==DialogResult.OK)
   {
      txtEditor.LoadFile(ofd.FileName,RichTextBoxStreamType.PlainText);
   }
}
```

5.3.4. Yazdırma İletişim Kutusu (PrintDialog)

PrintDialog; yazıcı seçme, kâğıt boyutlarını ayarlama veya yazdırmayı başlatma gibi işlemlerin yapılmasını sağlayan iletişim kutusudur. PrintDialog sınıfından bir nesne oluşturularak bu nesne ile iletişim kutusunun özellikleri kontrol edilir.

```
PrintDialog pd = new PrintDialog();
pd.ShowDialog();
```

12. Uygulama

PrintDialog

Burada Editör uygulaması kullanılarak Dosya menüsündeki Yazdır menü elemanına tıklandığında iletişim kutusu açılacaktır. İletişim kutusundaki Tamam butonuna tıklandığında RichTextBox kontrolü içindeki metinleri yazdırma işlemi gerçekleştirilecektir.

1. Adım: PrintDialog sınıfından oluşturulan nesne iletişim kutusunu göstererek yazıcı seçimi ve ayarların yapılması sağlanır. Yazdırma işlemi için PrintDocument nesnesi kullanılacaktır. Tasarım görünümünde iken ToolBox penceresinden PrintDocument kontrolünü form içine ekleyiniz. PrintDocument kontrolünün name değerini belge olarak değiştiriniz.

2. Adım: Dosya menüsündeki Yazdır menü elemanı Click olayında PrintDialog nesnesi kullanılarak RichTextBox içindeki yazıların yazıcıya gönderilmesi işlemini yapan kodlamaları yazınız.

```
private void menuYazdir_Click(object sender, EventArgs e)
{
   PrintDialog pd = new PrintDialog();
   DialogResult cevap = pd.ShowDialog();
   if (cevap == DialogResult.OK)
   {
      belge.Print();
   }
}
```

3. Adım: Yazdırma işlemini başlatmak için form içine eklenen PrintDocument kontrolü PrintPage olayına aşağıdaki kodlamaları yazınız.

```
private void belge_PrintPage(object sender, PrintPageEventArgs e)
{
  e.Graphics.DrawString(txtEditor.Text, txtEditor.Font,
  Brushes.Black,new Point(100,100));
}
```

5.3.5. Yazı Tipi İletişim Kutusu (FontDialog)

FontDialog; yazı tipinin, büyüklüğünün ve renginin seçilmesine olanak sağlayan iletişim kutusudur.FontDialog sınıfından bir nesne oluşturularak bu nesne ile iletişim kutusunun özellikleri kontrol edilir.

13. Uygulama

FontDialog

Editör uygulamasında FontDialog sınıfında üretilen bir nesne kullanılarak RichTextBox içinde seçilen yazının tipini ve büyüklüğünü değiştirme işlemi yapılacaktır.

1. Adım: Editör uygulaması menülerine Biçim adında menü ekleyiniz. Biçim menüsü içine de Yazı Tipi adında açılır menü elemanını ekleyiniz.

2. Adım: Yazı Tipi açılır menüsünün name değerini menuYaziTipi olarak veriniz. Yazı Tipi alt menüsü Click olayında FontDialog iletişim kutusunu göstererek belirlenen yazı tipini ve boyutunu RichTextBox kontrolüne uygulayan kodlamayı yapınız.

```
private void menuYaziTipi_Click(object sender, EventArgs e)
{
  FontDialog fd = new FontDialog();
  if (fd.ShowDialog() == DialogResult.OK)
  {
    txtEditor.Font = fd.Font;
  }
}
```

Sıra Sizde

1. Uygulamada ShowDialog metodu neden karşılaştırma ifadesi içinde kullanıldı? Açıklayınız.
2. Uygulamada RichTextBox içindeki metnin sadece seçim yapılan kısımlarının biçimini değiştiren kodlamayı yapınız.

5.3.6. Renk İletişim Kutusu (ColorDialog)

ColorDialog, renk paleti içinden bir renk seçilmesini sağlayan iletişim kutusudur.

14. Uygulama

ColorDialog

Editör uygulamasında ColorDialog sınıfında üretilen bir nesne kullanılarak iletişim kutusu içinden seçilen bir rengin RichTextBox kontrolüne uygulanması işlemi yapılacaktır.

1. Adım: Editör uygulaması Biçim menüsü içine Renk adında açılır menü ekleyiniz. Renk menü elemanının name değerini menuRenk olarak veriniz.

2. Adım: Renk menü elemanı Click olayında RichTextBox içinden seçilen metnin renginin ColorDialog iletişim kutusu ile değiştirilmesini sağlayan kodlamayı yapınız.

```
private void menuRenk_Click(object sender, EventArgs e)
{
  ColorDialog cd = new ColorDialog();
  if (cd.ShowDialog() == DialogResult.OK)
  {
    txtEditor.SelectionColor = cd.Color;
  }
}
```

Sıra Sizde

ColorDialog iletişim kutusunun yaptığı görevi, FontDialog iletişim kutusu da yapabilir. FontDialog içinden renk seçme işlemi için gerekli düzenlemeleri yapınız.

5.4. VERİ DOĞRULAMA (VALIDATION)

Veri girişi ile işlem yapılan uygulamalarda önemli noktalardan biri de verinin doğru girilmesidir. Doğru veri girilmemesi veya eksik bilgi girişi yapılması durumunda uygulamada hatalar, aksaklıklar ortaya çıkacak veya uygulama çalışamaz hâle gelecektir.

Yazılımcı, bir uygulamayı geliştirirken bu uygulamayı kullanacak kişilerin yapabileceği hataları öngörerek önlem almalıdır. Örneğin öğrenci not girişleri yapılan uygulamada sınav notu 100 olan bir öğrenciye yanlışlıkla 1000 notu girilirse öğrencinin ortalaması çok yüksek çıkacaktır veya rakam yerine bir harf girilirse uygulama hata verecektir. Uygulama geliştirilirken kullanıcılar bilgilendirilerek ve hatalara karşı önlem alınarak oluşabilecek problemlerin önüne geçilir.

5.4.1. İpucu (ToolTip)

ToolTip, form içindeki nesnelere ait bilgilerin görüntülendiği açıklama pencereleridir. Form içindeki bir nesne üzerine fare ile gelindiğinde açıklama bilgisinin görüntülenmesini ToolTip sağlar.

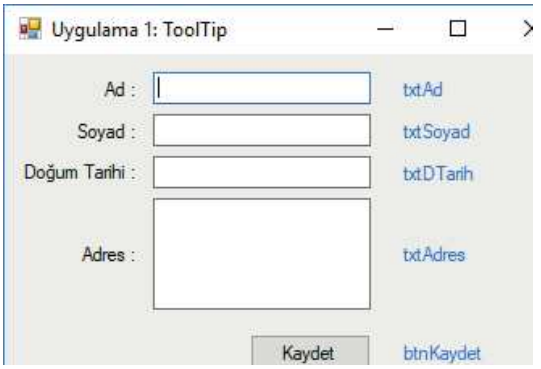
15. Uygulama

ToolTip

Bu uygulama, ToolTip sınıfından üretilen nesne ile form içindeki kontrollerin üzerine gelindiğinde açıklama pencerelerinin gösterilmesidir.

1. Adım: Görsel 5.15'teki form tasarımını yapınız ve form içindeki kontrollere name değerlerini veriniz.

2. Adım: Formun Load olayında ToolTip sınıfından bir nesne oluşturarak form içindeki kontrollerde açıklama pencerelerini gösteren kodlamayı yapınız.



Görsel 5.15: ToolTip uygulaması form tasarımı

```
private void Form1_Load(object sender, EventArgs e)
{
  ToolTip tt = new ToolTip();
  tt.SetToolTip(txtAd, "Adınızı giriniz.");
  tt.SetToolTip(txtSoyad, "Soyadınızı giriniz.");
  tt.SetToolTip(txtDTarih, "Doğum tarihinizi gg/aa/yyyy şeklinde giriniz.");
  tt.SetToolTip(txtAdres, "Adresinizi giriniz.");
  tt.SetToolTip(btnKaydet, "Kaydetmek için tıklayınız.");
}
```

5.4.2. Veri Girişi Doğrulama (Input Validation)

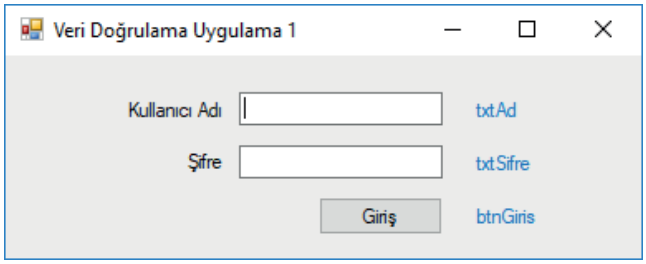
Veri doğrulama, uygulamanın her katmanında yapılabilir. Yazılımcı tarafından belirlenen bazı ölçütlere göre kullanıcıların girdiği bilgilerin doğruluğu kontrol edilmelidir. Kullanıcının girdiği bilgiler ilk olarak form seviyesinden kontrol edilir. Form uygulamalarında veri girişleri en çok TextBox kontrolleri ile gerçekleşir. Yazılımcı TextBox'a girilen verilerin doğruluğunu kullanıcının klavyeden girdiği her karakter esnasında sağlayabileceği gibi Validating olayı ile de kontrol edebilir. Girilen veriler uygun biçimde olmadığında uyarı mesajları verilir veya ErrorProvider nesnesi kullanılarak bilgilendirme işlemi gerçekleştirilir.

16. Uygulama

Veri Girişi Doğrulama

Burada form içinden veri girişi yapılan alanlara belirlenen ölçütlere göre doğru veri girişi yapılmadığı takdirde uyarı mesajı veren uygulama gerçekleştirilecektir.

1. Adım: Görsel 5.16'daki form tasarımını yapınız ve form içindeki kontrollere name değerlerini veriniz.



Görsel 5.16: Veri doğrulama uygulaması form tasarımı

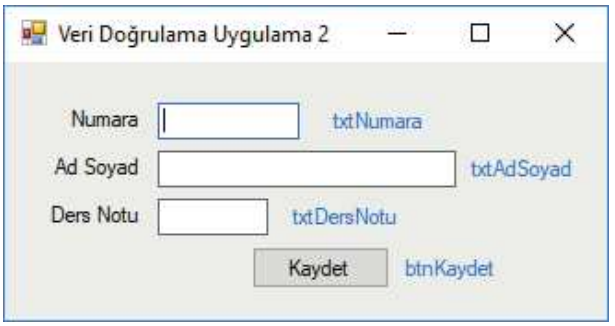
2. Adım: Giriş butonu Click olayında kullanıcı adı ve şifre verileri girilmez ise kullanıcıyı mesaj ile uyaran kodlamaları yapınız.

```
private void btnGiris_Click(object sender, EventArgs e)
{
  if (string.IsNullOrEmpty(txtAd.Text))
  {
     MessageBox.Show("Kullanıcı adını giriniz.", "Uyarı");
  }
  if(string.IsNullOrEmpty(txtSifre.Text))
  {
     MessageBox.Show("Şifreyi giriniz.", "Uyarı");
  }
}
```

17. Uygulama

Bu uygulamada uyarı mesajları MessageBox yerine ErrorProvider nesnesi kullanılarak verilecektir. Doğrulama kontrolü ise TextBox nesnelerinin Validating olayı içinde gerçekleştirilecektir.

1. Adım: Görsel 5.17'deki form tasarımını yapınız ve kontrollere name değerlerini veriniz.



Görsel 5.17: Veri doğrulama uygulaması form tasarımı

2. Adım: Uygulamada ErrorProvider sınıfından üretilen nesneyi global olarak oluşturunuz.

```
ErrorProvider ep = new ErrorProvider();}
```

3. Adım: Form içindeki txtNumara isimli TextBox kontrolüne sayısal bir değer girilmelidir. TextBox içine girilen değerin sayısal olup olmadığını Validating olayında kontrol eden kodlamaları yapınız.

```
private void txtNumara_Validating(object sender, CancelEventArgs e)
{
  if(int.TryParse(txtNumara.Text,out int sonuc))
  {
     ep.SetError(txtNumara, "");
  }
  else
  {
     e.Cancel = true;
     ep.SetError(txtNumara, "Numara girişi hatalı");
  }
}
```

4. Adım: txtAdSoyad isimli TextBox kontrolünün Validating olayında TextBox içine değer girilmemiş ise ErrorProvider nesnesi ile kullanıcıyı uyaran doğrulama kodlarını yazınız.

```
private void txtAdSoyad_Validating(object sender, CancelEventArgs e)
{
   if (txtAdSoyad.Text == "")
   {
      e.Cancel = true;
      ep.SetError(txtAdSoyad, "Adı ve soyadı giriniz.");
   }
   else
   {
      ep.SetError(txtAdSoyad, "");
   }
}
```

5. Adım: txtDersNotu isimli TextBox kontolünün Validating olayında TextBox içine girilen değer sayısal değilse veya sayısal olup 0 ile 100 arasında değilse ErrorProvider nesnesi ile kullanıcıyı uyaran doğrulama kodlarını yazınız.

```
private void txtDersNotu_Validating(object sender, CancelEventArgs e)
{
  int dersNotu;
  if(int.TryParse(txtDersNotu.Text,out dersNotu))
 {
      if(dersNotu<0 || dersNotu>100)
      {
         e.Cancel = true;
         ep.SetError(txtDersNotu, "0-100 arasında değer giriniz.");
      }
      else
      {
         ep.SetError(txtDersNotu, "");
      }
   }
   else
   {
      e.Cancel = true;
      ep.SetError(txtDersNotu, "Sayısal değer giriniz.");
   }
}
```

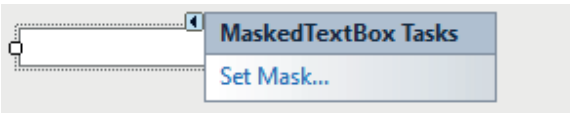
Sıra Sizde

Uygulamada e.Cancel = true ifadesi neden kullanılmıştır? Düşüncelerinizi sınıf arkadaşlarınızla paylaşınız.

5.4.3. Veri Girişi Maskeleme (MaskedTextBox)

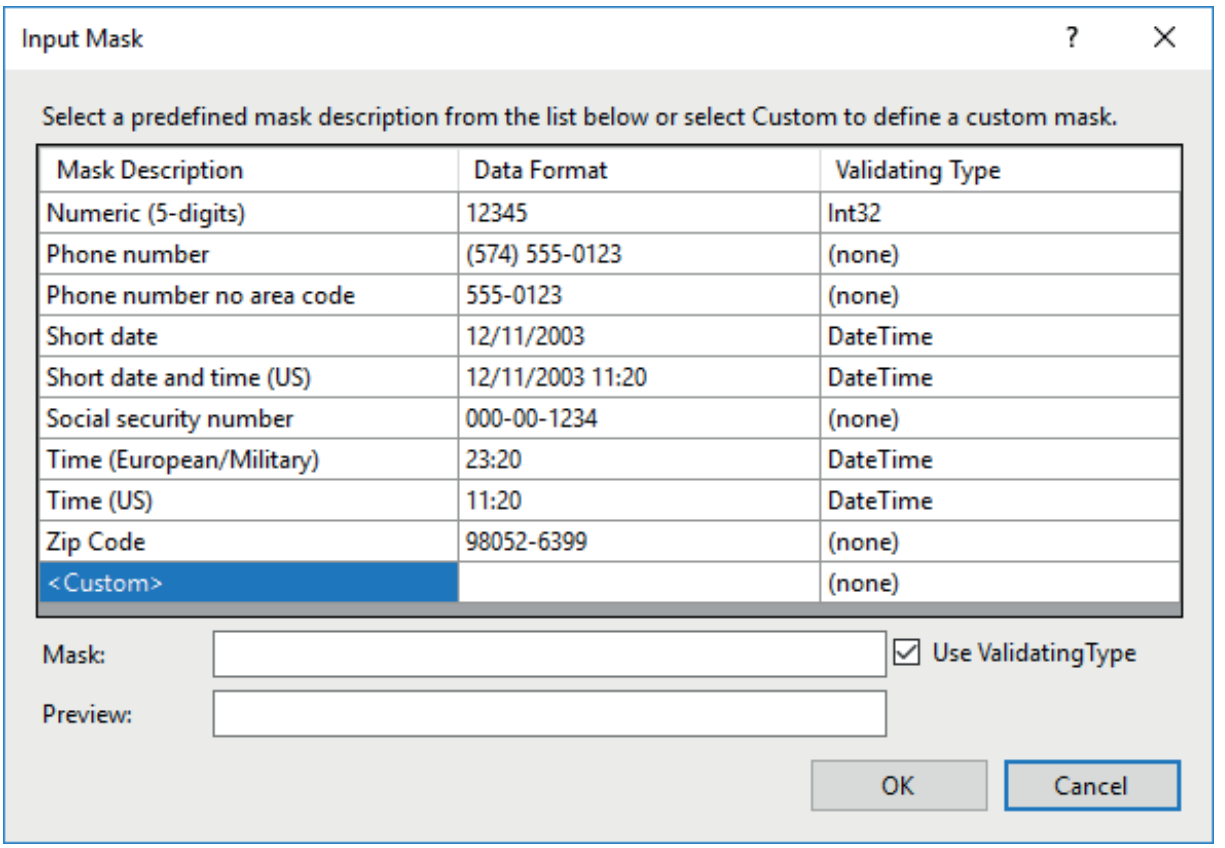
MaskedTextBox, belirlenen formatta veri girişinin yapılmasını sağlayan bir kontroldür. MaskedTextBox kontrolleri, form içinde kullanıcının gireceği verilerin doğrulanmasını sağlar. Örneğin telefon numarası girişi yapılan bir alana uygun formatta veri girilmesi için MaskedTextBox kontrolü kullanılır.

Forma eklemek için tasarım ekranında iken Toolbox penceresinden MaskedTextBox kontrolü form içine sürüklenerek bırakılabilir. MaskedTextBox kontrolü eklendikten sonra giriş maskesi için Görsel 5.18'de gösterilen **Set Mask** linkine veya Properties penceresinden **Mask** özelliğine tıklanarak açılan diyalog kutusundan maske formatı belirlenebilir.



Görsel 5.18: Giriş maskesi linki

Görsel 5.19'da açılan diyalog kutusundan maske formatlarının listesi görülebilir ve istenilen bir format seçilerek, giriş maskesi olarak kullanılabilir.



Görsel 5.19: Giriş maskesi diyalog penceresi

Bazı durumlarda Input Mask diyalog kutusundaki formatlar dışında özel formatlar belirlemek için Mask alanına maskeleme karakterleri girilerek özel maskeler oluşturulabilir. Bu karakterler şunlardır:

0	-	Rakam (Girilmesi zorunludur.)
9	-	Rakam veya boşluk (Girilmesi isteğe bağlıdır.)
#	-	Rakam veya boşluk (Girilmesi isteğe bağlıdır. Maskede bu konum boşsa özellik de bir boşluk olarak işlenir. Artı ve eksi işaretlerine izin verilir.)
L	-	Harf (Girilmesi zorunludur. A-Z veya a-z arasındaki harflere izin verir.)
?	-	Harf (Girilmesi isteğe bağlıdır. A-Z veya a-z arasındaki harflere izin verir.)
&	-	Karakter (Girilmesi zorunludur.)
C	-	Karakter (Girilmesi isteğe bağlıdır.)
A	-	Hem harf hem de rakam (Girilmesi zorunludur.)
a	-	Hem harf hem de rakam (Girilmesi isteğe bağlıdır.)
.	-	Ondalık ayracı
,	-	Binlik ayracı
:	-	Saat ayracı
/	-	Tarih ayracı
$	-	Para birimi işareti
>	-	Tüm karakterleri küçük harfe çevirir.
<	-	Tüm karakterleri büyük harfe çevirir.

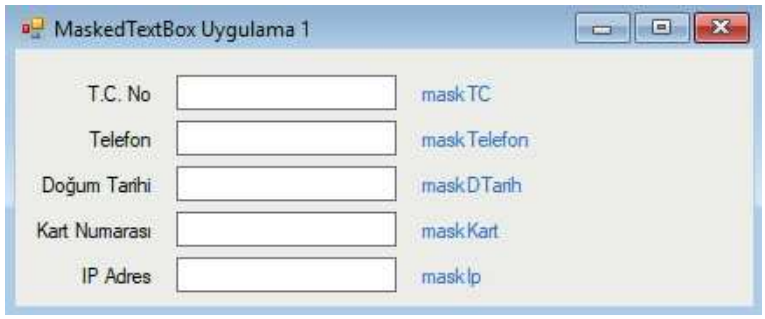
18. Uygulama

MaskedTextBox

Bu uygulamada MaskedTextBox kontrollerine Form Load olayı içinde farklı maskeler verilecektir.

1. Adım: Görsel 5.20'deki form tasarımını yapınız ve kontrollere name değerlerini veriniz.

2. Adım: Form Load olayında ilgili MaskedTexBox kontrollerine aşağıdaki kodlamaları yaparak maske formatlarını veriniz.



Görsel 5.20: Giriş maskeleme uygulaması

```
private void Form7_Load(object sender, EventArgs e)
{
  maskTC.Mask = "00000000000";
  maskTelefon.Mask = "(999) 000 00 00";
  maskDTarih.Mask = "00/00/0000";
  maskKart.Mask = "0000-0000-0000-0000";
  maskIp.Mask = "###.###.###.###";
}
```

5.5. VERİ BAĞLAMA (DATA BINDING)

Veri bağlama, veri kaynağının form içindeki bir kontrole bağlanmasıdır. Veri bağlama işleminde form kontrolü ile kaynak arasında veri gönderme veya alma yapılabilir. Veri bağlama en çok veri tabanı içinde depolanan verilerle işlem yapılırken kullanılsa da kontrollerle, dizilerle veya koleksiyonlarla da veri bağlama işlemi yapılabilir.

5.5.1. Basit Veri Bağlama (Simple Data Binding)

Basit veri bağlama, bir form kontrolüne tek veri bilgisinin bağlanmasıdır. Basit veri bağlama genellikle veri kümesi içindeki bir veriyi form içindeki kontrole aktarmak için kullanılır. Basit veri bağlama işlemlerinde Binding sınıfında üretilen nesne kullanılır. Bu nesne temelde üç parametre alır. Bu parametreler şunlardır:

```
Binding binding=new Binding(string propertyName, object dataSource, string dataMember )
```

Binding sınıfındaki parametrelerin işlevleri aşağıda verilmiştir.

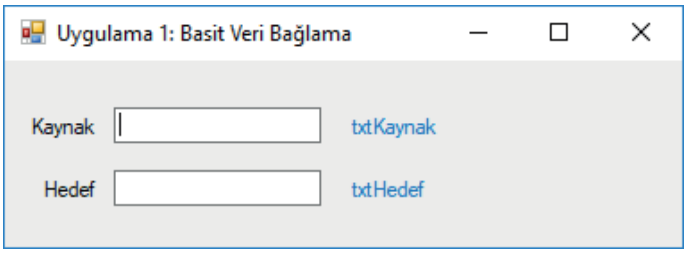
- propertyName: Bağlanacak olan nesnenin özelliğini belirtir.
- dataSource: Veri kaynağını belirtir.
- dataMember: Veri kaynağının hangi özelliğinin bağlanacağını belirtir.

19. Uygulama

Basit Veri Bağlama

Bu uygulamada form içindeki iki TextBox kontrolünden biri veri kaynağı olarak kullanılacak ve diğer TextBox kontrolüne veri bağlama işlemi gerçekleştirilecektir.

1. Adım: Görsel 5.21'deki form tasarımını yapınız ve kontrollere name değerlerini veriniz.



Görsel 5.21: Basit veri bağlama uygulaması

2. Adım: Formun Load olayına veri bağlama işlemleri için kullanılan kodlamaları yapınız.

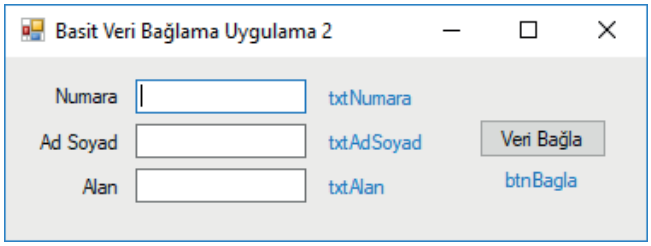
```
private void Form1_Load(object sender, EventArgs e)
{
  Binding bagla = new Binding("Text", txtKaynak, "Text");
  txtHedef.DataBindings.Add(bagla);
}
```

20. Uygulama

Basit Veri Bağlama

Bu uygulamada bir sınıf oluşturulacaktır. Sınıftan üretilen nesne ile sınıf özeliklerine değer aktarılarak form içinde kontrollere veri bağlama işlemi gerçekleştirilecektir.

1. Adım: Görsel 5.22'deki form tasarımını yapınız ve kontrollere name değerlerini veriniz.



Görsel 5.22: Basit veri bağlama uygulaması

2. Adım: Veri kaynağı olarak kullanmak için aşağıda özellikleri verilen sınıfı oluşturunuz.

```
class Ogrenciler
{
    public int Numara { get; set; }
    public string AdSoyad { get; set; }
    public string Alan { get; set; }
}
```

3. Adım: Veri Bağla butonu Click olayında sınıf özelliklerine değer aktarıldıktan sonra TextBox kontrollerine veri bağlama işlemini gerçekleştiren kodlamaları yapınız.

```
private void btnBagla_Click(object sender, EventArgs e)
{
   Ogrenciler ogrenci = new Ogrenciler();
   ogrenci.Numara = 1111;
   ogrenci.AdSoyad = "Mehmet";
   ogrenci.Alan = "Bilişim Teknolojileri";
   txtNumara.DataBindings.Add("Text", ogrenci, "Numara");
   txtAdSoyad.DataBindings.Add("Text", ogrenci, "AdSoyad");
   txtAlan.DataBindings.Add("Text", ogrenci, "Alan");
}
```

5.5.2. Kompleks Veri Bağlama (Complex Data Binding)

Basit veri bağlama, bir kontrol nesnesine tek bir veri değerinin bağlanması işlemidir. Kompleks veri bağlama işlemi ise bir kontrol nesnesine birden çok veri değerinin bağlanması işlemidir. ListBox, ComboBox ve DataGridView kontrolleri, kompleks veri bağlanmasında en çok kullanılan kontrollerdir.

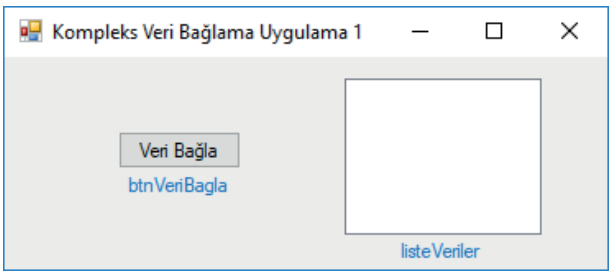
Kompleks bağlamada veri bağlanacak olan kontrolün **DataSource** özelliği ile veri kaynağı belirtilerek bağlama işlemi gerçekleştirilir.

21. Uygulama

Kompleks Veri Bağlama

Bu uygulamada dizideki verileri ListBox kontrolü içinde gösterme işlemi yapılacaktır. Diziler konusunda döngülerle gerçekleştirilen işlem, bu uygulamada veri bağlama yöntemi ile gerçekleştirilecektir.

1. Adım: Görsel 5.23'teki gibi form tasarımını yapınız ve kontrollere name değerlerini veriniz.



Görsel 5.23: Kompleks veri bağlama uygulaması form tasarımı

2. Adım: Veri Bağla butonu Click olayında bir dizi oluşturarak dizinin değerlerini ListBox kontrolü içinde göstermek için veri bağlama işleminin kodlarını yazınız.

```
private void btnVeriBagla_Click(object sender, EventArgs e)
{
   string[] diller = { "C#", "Java", "Python","Delphi"};
   listeVeriler.DataSource = diller;
}
```

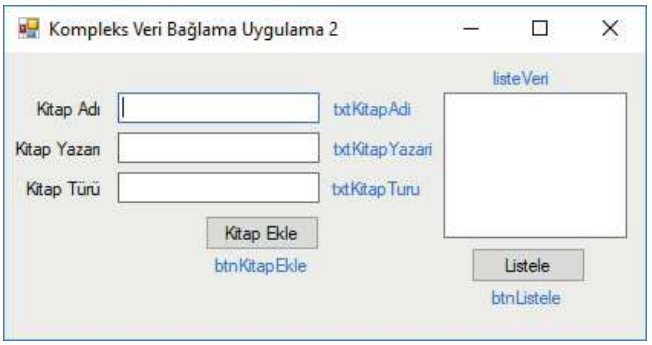
22. Uygulama

Kompleks Veri Bağlama

Bu uygulamada ArrayList koleksiyonu içine sınıf nesnesi eklenerek ListBox içinde gösterilecektir.

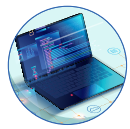
1. Adım: Görsel 5.24'teki form tasarımını yapınız ve kontrollere name değerlerini veriniz.

2. Adım: Kitaplar adında bir sınıf oluşturunuz ve bu sınıfın özelliklerini belirleyen kodlamaları yapınız.



Görsel 5.24: Kompleks veri bağlama uygulaması form tasarımı

```
class Kitaplar
{
    public string KitapAdi { get; set; }
    public string KitapYazari { get; set; }
    public string KitapTuru { get; set; }
}
```



3. Adım: Veri kaynağı şeklinde kullanılacak ArrayList koleksiyonunu global olarak oluşturunuz.

```
ArrayList kaynakVeri = new ArrayList();
```

4. Adım: Kitap Ekle butonu Click olayında Kitaplar sınıfından üretilen nesne kullanarak sınıfın özelliklerine değer aktarımını ve nesneyi ArrayList koleksiyonuna ekleme işlemini yapınız.

```
private void btnEkle_Click(object sender, EventArgs e)
{
   Kitaplar kitap = new Kitaplar();
   kitap.KitapAdi =txtKitapAdi.Text;
   kitap.KitapYazari = txtKitapYazari.Text;
   kitap.KitapTuru = txtKitapTuru.Text;
   kaynakVeri.Add(kitap);
}
```

5. Adım: Listele butonu Click olayında ListBox kontrolüne ArrayList nesnesini bağlayınız. ListBox kontrolüne bağlanan veriler içinden KitapAdi özelliğinin gösterilmesini sağlayan kodlamaları yapınız.

```
private void btnListele_Click(object sender, EventArgs e)
{
  listeVeri.DataSource=null;
  listeVeri.DataSource = kaynakVeri;
  listeVeri.DisplayMember = "KitapAdi";
}
```

Sıra Sizde

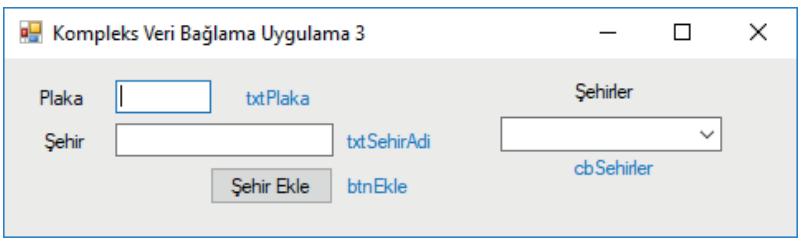
1. ListBox kontrolü içinde veri kaynağının diğer özelliklerinin gösterilmesini sağlayan kodlamaları yapınız.
2. Oluşturacağınız küçük gruplarla Listele butonu Click olayında DataSource özelliğine neden null değerinin aktarıldığını araştırınız. Elde ettiğiniz sonuçları diğer gruplarla paylaşınız.

23. Uygulama

Kompleks Veri Bağlama

Bu uygulamada ComboBox kontrolü ve List koleksiyonu kullanılarak veri bağlama işlemleri gerçekleştirilecektir. List koleksiyonuna plaka numarası ve şehir ismi özelliğine sahip sınıf nesnesi eklenerek ComboBox kontrolü içinde gösterme işlemi yapılacaktır.

1. Adım: Görsel 5.25'teki form tasarımını yapınız ve kontrollere name değerlerini veriniz.



Görsel 5.25: Kompleks veri bağlama uygulaması form tasarımı

2. Adım: List koleksiyonuna eklenecek nesneler için sınıf oluşturunuz ve bu sınıfın özelliklerini belirleyen kodlamaları yapınız.

```
class Sehirler
{
  public string Plaka { get; set; }
  public string SehirAdi { get; set; }
}
```

3. Adım: Kullanılacak List koleksiyonunu global olarak oluşturunuz.

```
List<Sehirler> listSehirler = new List<Sehirler>();
```

4. Adım: Şehir Ekle butonu Click olayında Sehirler sınıfından üretilen nesne kullanılarak sınıfın özelliklerine değer aktarımı yapıldıktan sonra List koleksiyonuna ekleme işleminin kodlarını yazınız.

```
private void btnEkle_Click(object sender, EventArgs e)
{
   Sehirler sehir = new Sehirler();
   sehir.Plaka = txtPlaka.Text;
   sehir.SehirAdi = txtSehirAdi.Text;
   listSehirler.Add(sehir);
   Bagla();
}
```

5. Adım: Bagla() ismindeki metot ile List koleksiyonunun ComboBox kontrolüne veri bağlama işlemini gerçekleştiriniz.

```
private void Bagla()
{
cbSehirler.DataSource = null;
cbSehirler.DataSource = listSehirler;
cbSehirler.DisplayMember = "SehirAdi";
cbSehirler.ValueMember = "Plaka";
}
```

24. Uygulama

Kompleks Veri Bağlama

Bu uygulamada DataGridView kontrolü ve List koleksiyonu kullanılarak veri bağlama işlemleri gerçekleştirilecektir. List koleksiyonuna öğrenci numarası, adı ve sınav notu özelliklerine sahip sınıf nesnesi eklenerek DataGridView kontrolü içinde gösterme işlemi yapılacaktır.

1. Adım: Görsel 5.26'daki form tasarımını yapınız ve kontrollere name değerlerini veriniz.

Görsel 5.26: Kompleks veri bağlama uygulaması form tasarımı

2. Adım: Form kod görünümüne geçerek Ogrenciler isminde bir sınıfın özelliklerini oluşturan kodlamayı yapınız.

```
class Ogrenciler
{
  public int Numara { get; set; }
  public string AdSoyad { get; set; }
  public int DersNotu { get; set; }
}
```

3. Adım: Ogrenciler sınıfından nesneleri saklayacak liste isminde bir List koleksiyonunu global olarak oluşturunuz.

```
List<Ogrenciler> liste = new List<Ogrenciler>();
```

4. Adım: Ekle butonu Click olayına Ogrenciler sınıfından üretilen nesne kullanılarak sınıfın özelliklerine değer aktarıldıktan sonra List koleksiyonuna ekleme işleminin kodlarını yazınız.

```
private void btnEkle_Click(object sender, EventArgs e)
{
   Ogrenciler ogrenci = new Ogrenciler();
   ogrenci.Numara = int.Parse(txtNumara.Text);
   ogrenci.AdSoyad = txtAdSoyad.Text;
   ogrenci.DersNotu =int.Parse(txtDersNotu.Text);
   liste.Add(ogrenci);
   Bagla();
}
```

5. Adım: Bagla() ismindeki metot ile List koleksiyonunu DataGridView kontrolüne veri bağlama işlemini gerçekleştiriniz.

```
private void Bagla()
{
   gridListe.DataSource = null;
   gridListe.DataSource = liste;
}
```

6. Adım: DataGridView kontrolü içindeki hücrelerin Double Click olayında çift tıklanan hücrenin bulunduğu satırdaki bilgileri, TextBox kontrollerine aktaran kodlamayı yapınız.

```
private void gridListe_CellDoubleClick(object sender,DataGridViewCellEventArgs e)
{
   txtNumara.Text = gridListe.CurrentRow.Cells[0].Value.ToString();
   txtAdSoyad.Text = gridListe.CurrentRow.Cells[1].Value.ToString();
   txtDersNotu.Text = gridListe.CurrentRow.Cells[2].Value.ToString();
}
```

Sıra Sizde

Form içine Sil isminde bir buton ekleyiniz. Sil butonu Click olayına DataGridView kontrolünün seçili satırındaki bilgileri silme işlemini gerçekleştiren kodlamayı yapınız.

25. Uygulama

Bu uygulama 24. Uygulamada List koleksiyonu kullanılarak yapılan veri bağlama işlemini DataTable nesnesi kullanılarak DataGridView kontrolü içinde gösterme işlemi yapılacaktır.

1. Adım: Uygulamada global olarak bir DataTable nesnesi oluşturunuz.

```
DataTable tablo = new DataTable();
```

2. Adım: Formun Load olayına DataTable nesnesinin sütunlarını oluşturma işlemini yapınız.

```
private void Form1_Load(object sender, EventArgs e)
{
    tablo.Columns.Add("Numara", typeof(int));
    tablo.Columns.Add("Ad Soyad", typeof(string));
    tablo.Columns.Add("Notu", typeof(int));
 }
```

3. Adım: Ekle butonun Click olayına da DataTable nesnesine veri ekleme işlemini yapınız.

```
private void btnEkle_Click(object sender, EventArgs e)
{
   int numara = int.Parse(txtNumara.Text);
   string adsoyad = txtAdSoyad.Text;
   int notu = int.Parse(txtDersNotu.Text);

   tablo.Rows.Add(numara, adsoyad, notu);

   Bagla();
}
```

4. Adım: Bagla() ismindeki metot ile DataTable nesnesini kullanarak DataGridView kontrolüne veri bağlama işlemini gerçekleştiriniz.

```
private void Bagla()
{
   gridListe.DataSource = tablo;
}
```

ÖLÇME VE DEĞERLENDİRME

A) Aşağıdaki cümlelerde boş bırakılan yerlere doğru sözcükleri yazınız.

1. Form uygulamaları ile bilgisayar ortamında çalışan kullanıcı etkileşimli ..... geliştirilebilir.

2. Bir Windows form projesinde uygulamanın hangi formdan başlayacağı Program.cs dosyasındaki Main fonksiyonu içindeki ..... metodu ile belirlenir.

3. Windows form sınıfı ..... isim uzayı içinde bulunur.

4. Aşağıda form sınıfı için verilen özellik, metot ve olayların işlevlerini tablodaki boşluklara yazınız.

ControlBox– CenterToScreen – Load – AcceptButton – Show – FormClosed – CancelButton – Hide

	Formun ekranın ortasında açılmasını sağlar.
	Form kapandığında çalışan olaydır.
	Formun üzerindeki büyültme, küçültme ve kapatma butonlarını gösterir veya gizler.
	Form açılırken çalışan olaydır.
	Form aktifken klavyeden enter tuşuna basıldığında belirlenen bir butonun tıklanma olayı gerçekleşir.
	Formu göstermek için kullanılan metottur.

5. Aşağıdakilerden hangisi Toolbox penceresi içindeki Containers sekmesinde <u>bulunmaz</u>?

A) GroupBox B) Panel C) TabControl

D) SplitContainer E) WebBrowser

6. Aşağıdakilerden hangisi MenuStrip içine eklenen nesnelerden biri <u>değildir?</u>

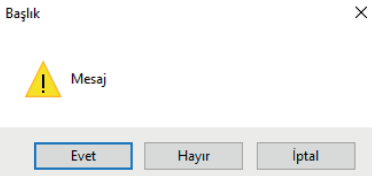
A) Label B) MenuItem C) ComboBox

D) Separator E) TextBox

7. Diyalog pencereleri ile ilgili aşağıda verilen bilgilerden hangisi <u>yanlıştır?</u>

A) OpenFileDialog, açılacak olan dosyanın konumunu belirler.
B) SaveFileDialog, kaydedilecek dosyanın konumunu belirler.
C) PrintDialog, yazdırma işlemini gerçekleştirir.
D) ColorDialog, renk paletinden renk seçilmesini sağlar.
E) FontDialog, yazı tipinin seçilmesini sağlar.

8. **Görseldeki gibi bir MessageBox penceresi oluşturmak için aşağıda verilen kodlardan hangisi Show() metodu içinde kullanılmalıdır?**



A) "Mesaj", "Başlık", MessageBoxButtons.YesNo, MessageBoxIcon.Error
B) "Mesaj", "Başlık", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Warning
C) "Başlık", "Mesaj", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Warning
D) "Başlık", "Mesaj", MessageBoxButtons.YesNo, MessageBoxIcon.Error
E) "Mesaj", " Başlık ", MessageBoxButtons.OKCancel, MessageBoxIcon. Warning

9. I. Girilmesi zorunlu rakamlar için kullanılır.
II. Para birimi işareti için kullanılır.
III. Tüm karakterleri büyük harfe çevirir.
IV. Girilmesi zorunlu harfler için kullanılır.

Yukarıda bilgileri verilen MaskedTextBox kontrolünde kullanılan maskeleme karakterleri için doğru sıralama aşağıdakilerden hangisidir?

A) #, $, < , L B) 9, $, < , A C) 0, $, < , a
D) 0, $, > , L E) 9, $, > , A

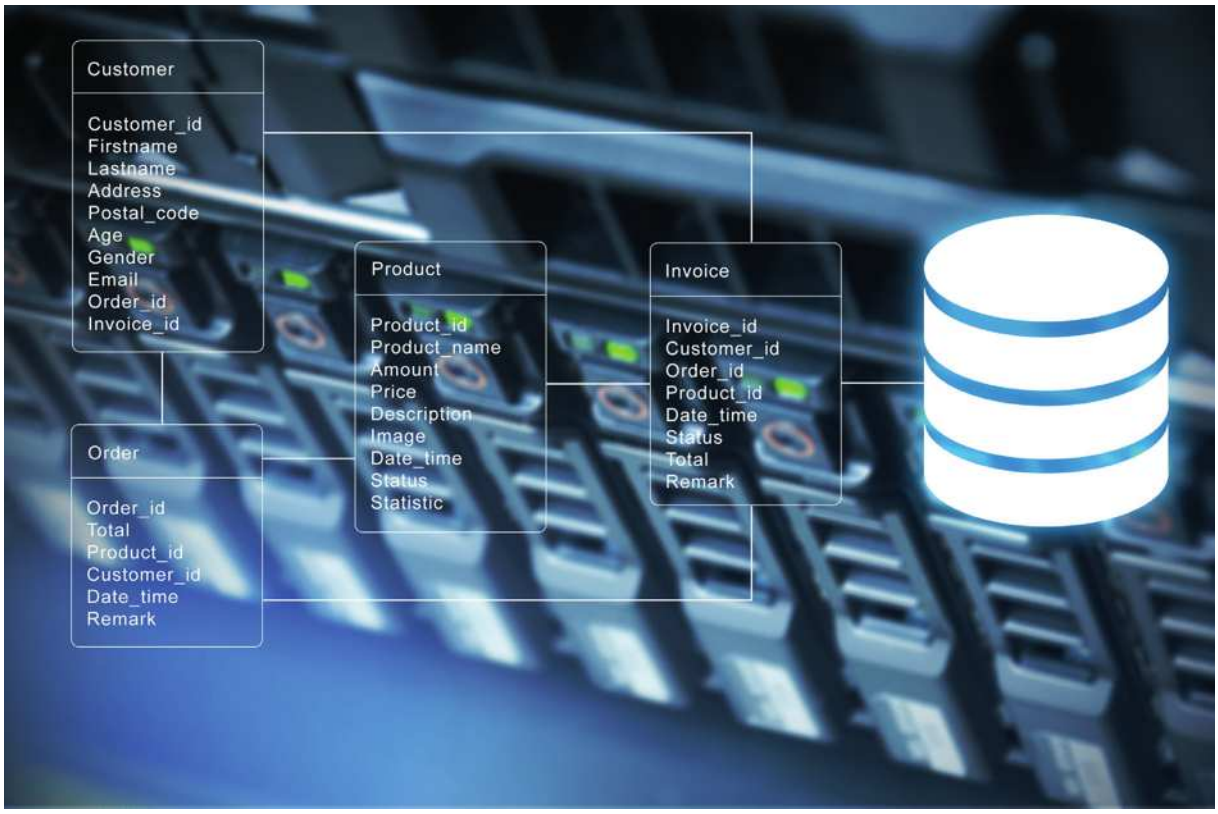
10. I. Tek veri için basit veri bağlama kullanılır.
II. Birden çok veri için kompleks veri bağlama kullanılır.
III. Basit veri bağlama için Binding sınıfından üretilen nesne kullanılır.
IV. ComboBox kontrolü kompleks veri bağlama için uygun değildir.
V. Kompleks veri bağlamada kontrolün DataSource özelliği kullanılır.

Yukarıdaki bilgilerden hangileri doğrudur?

A) I-II B) I-II-III C) I-II-V
D) I-II-III-IV E) I-II-III-V

6.
ÖĞRENME BİRİMİ

VERİ TABANI İŞLEMLERİ



KONULAR

6.1. VERİ TABANI YAZILIMI KURULUMU
6.2. VERİ TABANI OLUŞTURMA
6.3. TABLOLAR VE ÖZELLİKLER
6.4. SQL KOMUTLARI

ANAHTAR KELİMELER

Veri tabanı, MySQL, Veri tabanı tabloları, İlişkisel veri tabanı, SQL komutları, veri tabanı bağlantısı, CRUD işlemleri, Setup, Entity Framework, ORM

NELER ÖĞRENECEKSİNİZ?

- Veri tabanı
- Veri tabanı yönetim yazılımı
- Yapılandırılmış sorgu dili (SQL)
- Veri tabanı ve tablolar
- Veri türleri
- Tabloları ilişkilendirme
- SQL komutları
- Veri tabanı ile nesne tabanlı programlama arasında bağlantı
- Nesne tabanlı programlama üzerinden SQL komutları
- Entity Framework yapısı
- Projeye Setup hazırlama ve bilgisayara yükleme

HAZIRLIK ÇALIŞMALARI

1. Günlük hayatta kullanılan uygulamalarda veya web sitelerinde verilerin nasıl saklandığını araştırınız.
2. Veri tabanı türlerinin birbirlerine göre avantajlarını ve dezavantajlarını araştırınız.

6.1. VERİ TABANI YAZILIMININ KURULUMU

Verilerin kalıcı olarak saklanması için programlar, veri tabanı adı verilen yapılarla birlikte kullanılmalıdır. Bunun için öncelikle veri tabanı yapısı, tablo yapısı ve SQL sorgularının kullanımı bilinmelidir. Ardından form uygulaması ile veri tabanı arasında bağlantı kurularak, verilerin kalıcı olarak depolandığı uygulamalar geliştirilebilir.

6.1.1. Veri Tabanı (Database)

Verileri sistemli bir şekilde içinde barındıran yapılara **veri tabanı** adı verilir. Veri tabanı bir veya birden fazla tablodan oluşur. Veriler bu tablolarda saklanır. Gerektiğinde kullanıcı tarafından tablolardaki bu verilere erişilebilir ve veriler üzerinde değişiklik yapılabilir (Görsel 6.1).

Görsel 6.1: Veri tabanı işlemleri

Günümüzdeki uygulamaların tamamına yakınında verilerin saklanması için veri tabanı kullanılır. Örneğin ürün stok takip programları, muhasebe yazılımları, sağlık bilgi sistemleri, e-Ticaret yazılımları, e-Devlet, e-Okul Sistemi gibi uygulamalarda verilerin saklanması için veri tabanı kullanılır. Bu uygulamalardaki verilere istendiği zaman ulaşılabilmesinin nedeni, verilerin veri tabanında kayıtlı olmasıdır. Günlük hayatta sıklıkla karşılaşılan sosyal medya uygulamalarının tamamında da veri tabanı kullanılır. Paylaşılan içeriklere, erişim yetkisi olan kişiler tarafından istendiği zaman ulaşılabilir.

Verileri saklamak için metin dosyaları da kullanılabilir fakat bu dosyalar özellikle veriler büyüdükçe kontrol edilemez bir hâl alır. Bu yüzden verilerin saklanmasında fazla tercih edilmez.

6.1.2. Veri Tabanı Yönetim Sistemi (Database Managament System)

Veri tabanları genel olarak **VTYS** adı verilen uygulamalar ile kullanılır. Bu uygulamalar kullanıcıya; veri tabanı oluşturma, tablo oluşturma, tablolar arası ilişkiler kurma, veri ekleme, silme, güncelleme ve listeleme gibi işlemleri yaparken kolaylık sağlar. Ayrıca yedekleme, geri yükleme, veri tabanı ile bağlantı kurma, veri tabanı bakımı ve veri tabanı güvenliğini sağlama gibi yönetimsel işlemler de bu programlar sayesinde kolaylıkla yapılabilir.

En çok kullanılan VTYS yazılımlarına MySQL, SQL Server, Oracle, SQLite, Postgre SQL, IBM DB2 gibi uygulamalar örnek verilebilir. Bu öğrenme biriminde açık kaynak kodlu ve ücretsiz bir sistem olduğu için MySQL veri tabanı kullanılacaktır.

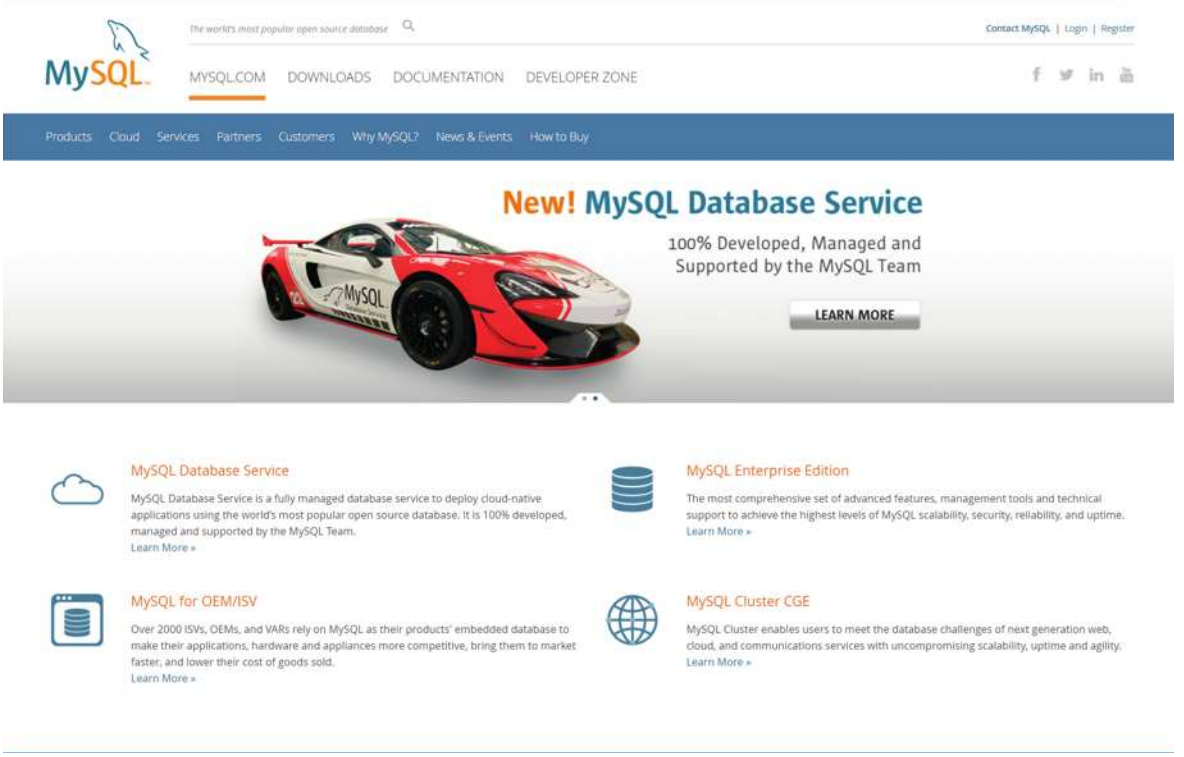
Görsel 6.2: Veri tabanı yönetimi

Sıra Sizde

1. VTYS yazımlarının birbirlerine göre avantajlarını ve dezavantajlarını araştırınız.
2. İçinde çok büyük veriler barındıran e-Devlet, sosyal medya vb. uygulamaların bu verileri nasıl sakladığını araştırınız.

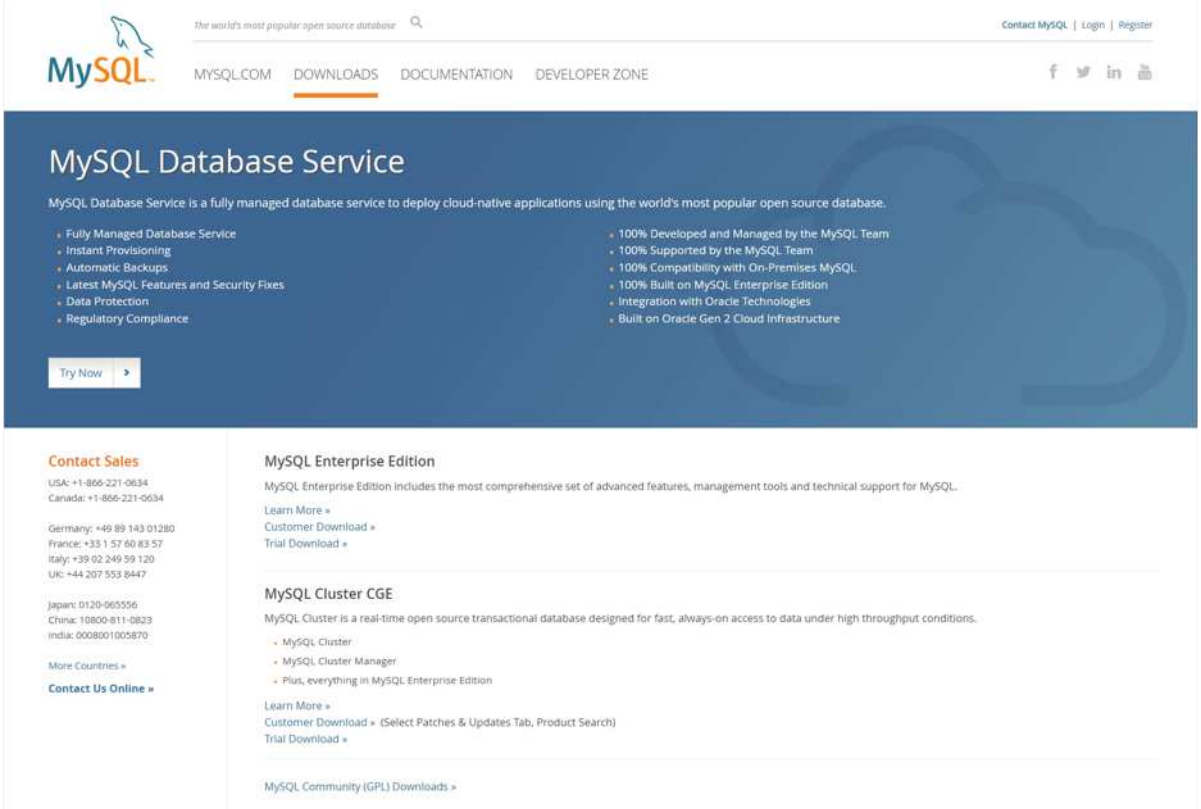
6.1.3. MySQL Veri Tabanı Yazılımının Kurulumu

MySQL resmî web sitesine girildiğinde Görsel 6.3'teki ekranla karşılaşılır. Üst tarafta yer alan DOWNLOADS menüsüne tıklanır.



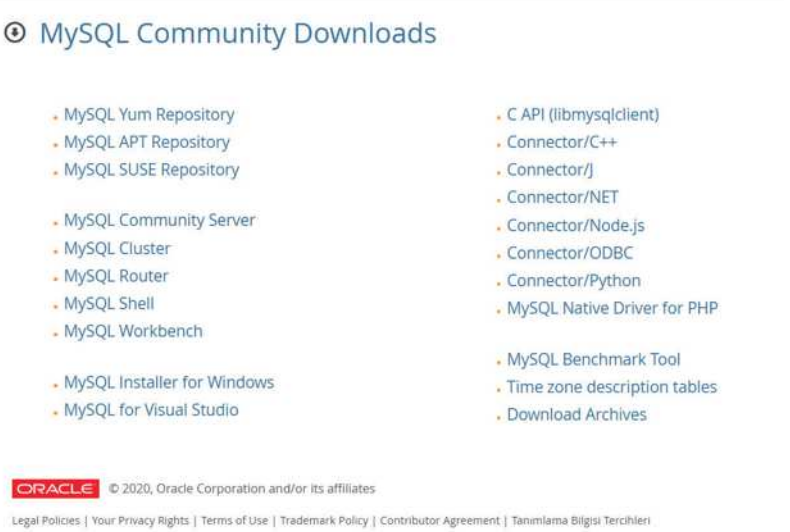
Görsel 6.3: mysql.com sayfası

Gelen sayfada en alttaki **"MySQL Community (GPL) Downloads"** kısmına tıklanır (Görsel 6.4).



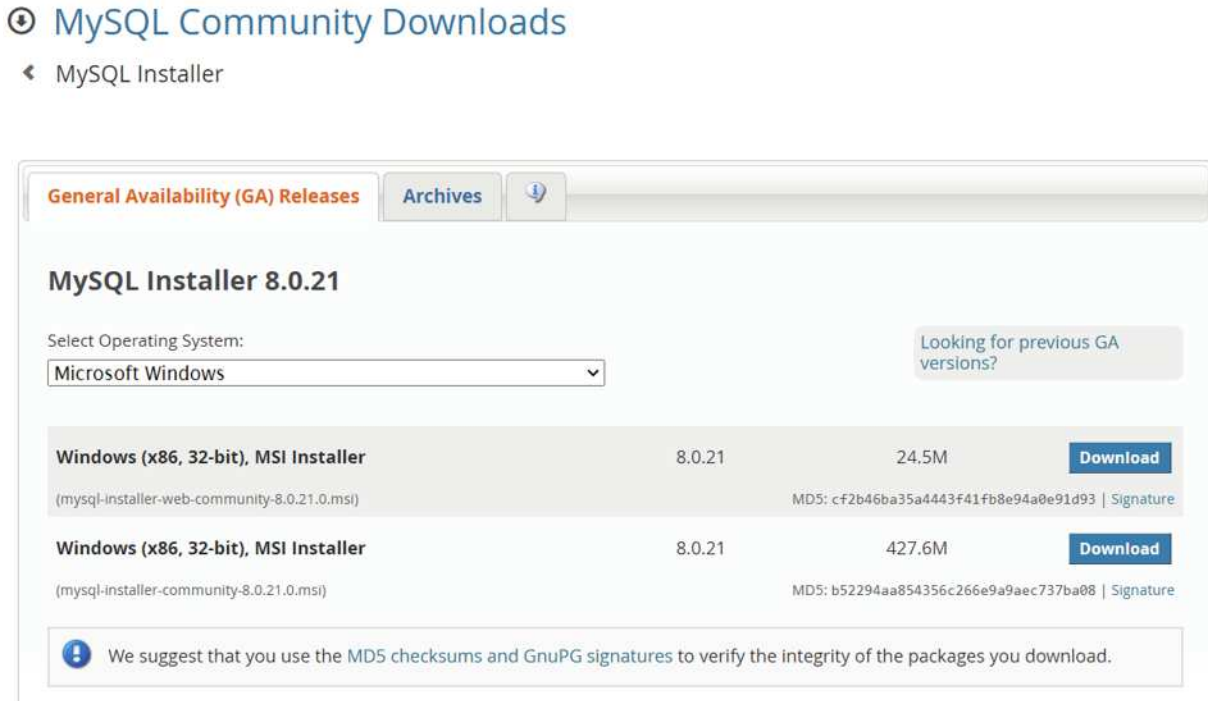
Görsel 6.4: mysql.com downloads sayfası

Gelen sayfadan uygun olan MySQL sürümü seçilir. Burada Windows işletim sistemine masaüstü (Desktop) uygulaması kurulacağı için **"MySQL Installer for Windows"** seçeneği seçilir (Görsel 6.5).



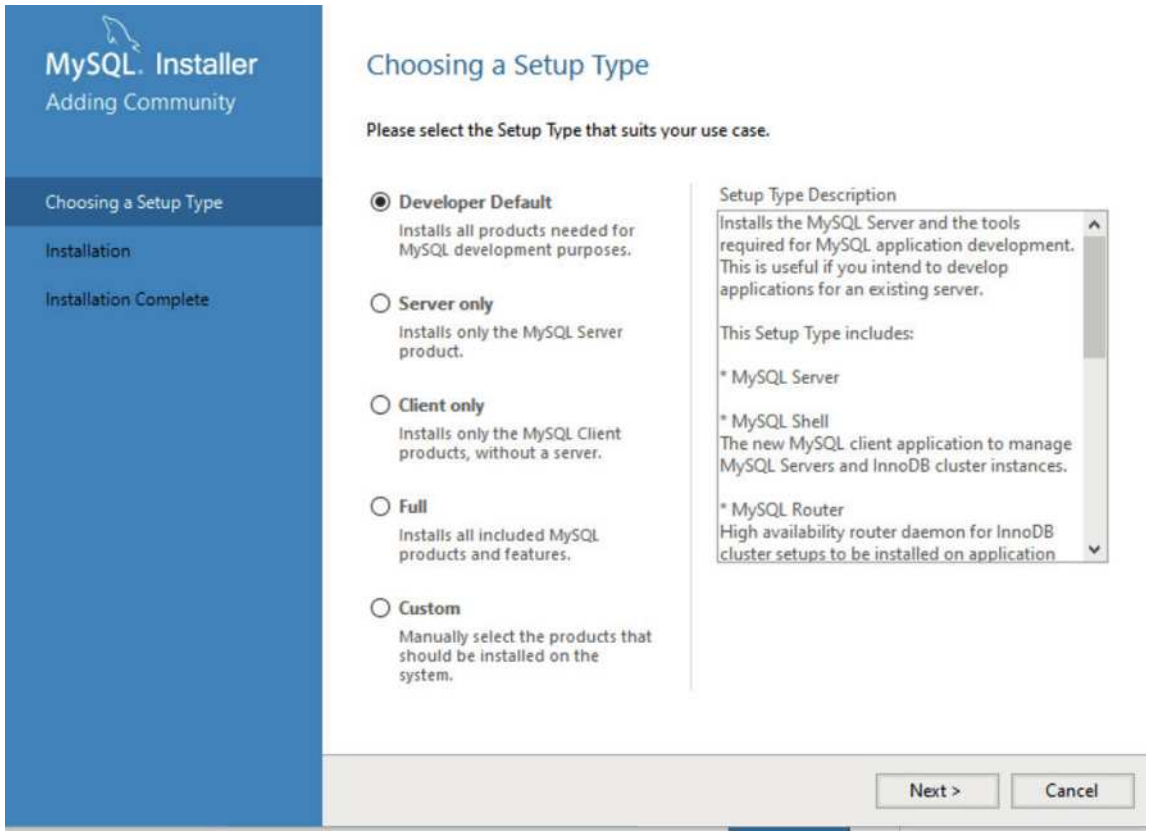
Görsel 6.5: MySQL Community Downloads sayfası

Görsel 6.6'daki sayfadan uygun versiyon seçilir. İndirme işlemi bittikten sonra kurulum adımlarına geçilir.



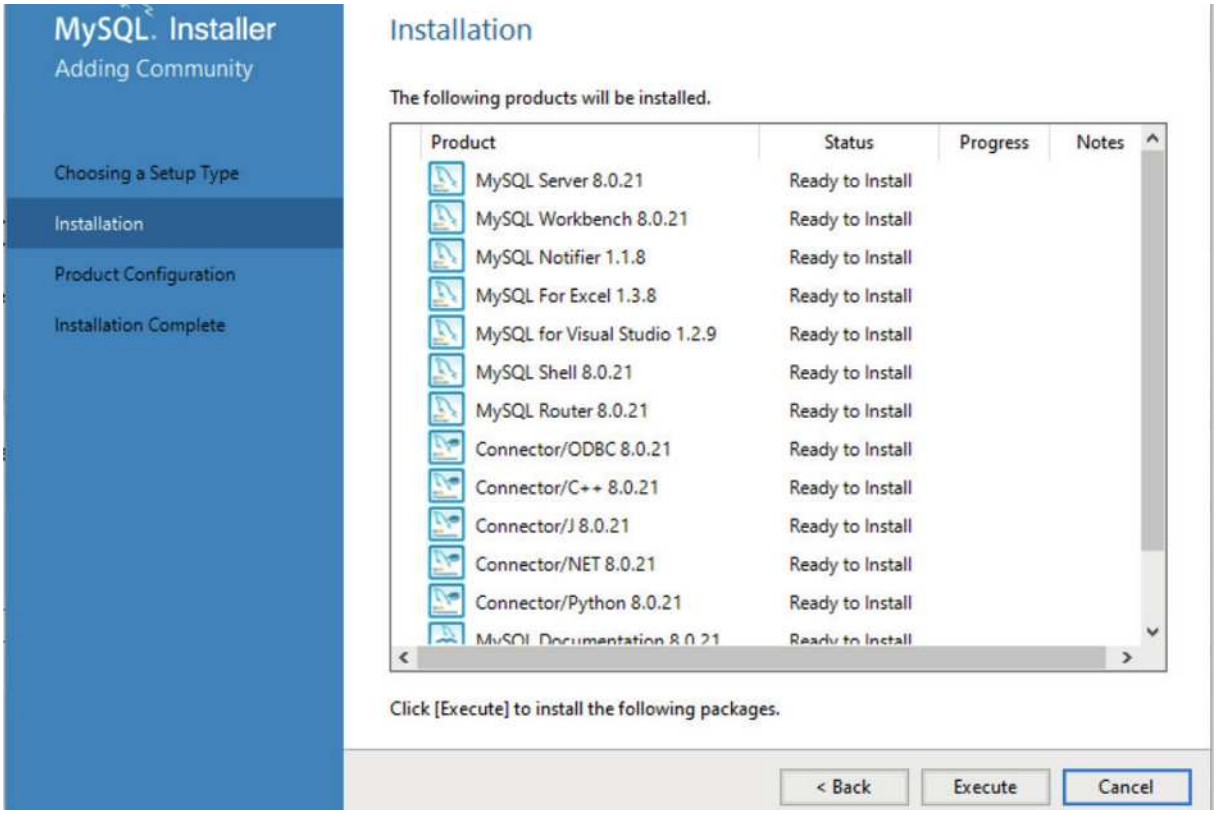
Görsel 6.6: Windows işletim sistemi için MySQL indirme sayfası

Görsel 6.7'deki ekranda kurulum türü seçilmelidir. Developer Default (Geliştirici Standart) seçeneği işaretlenip "Next" düğmesine basılır.

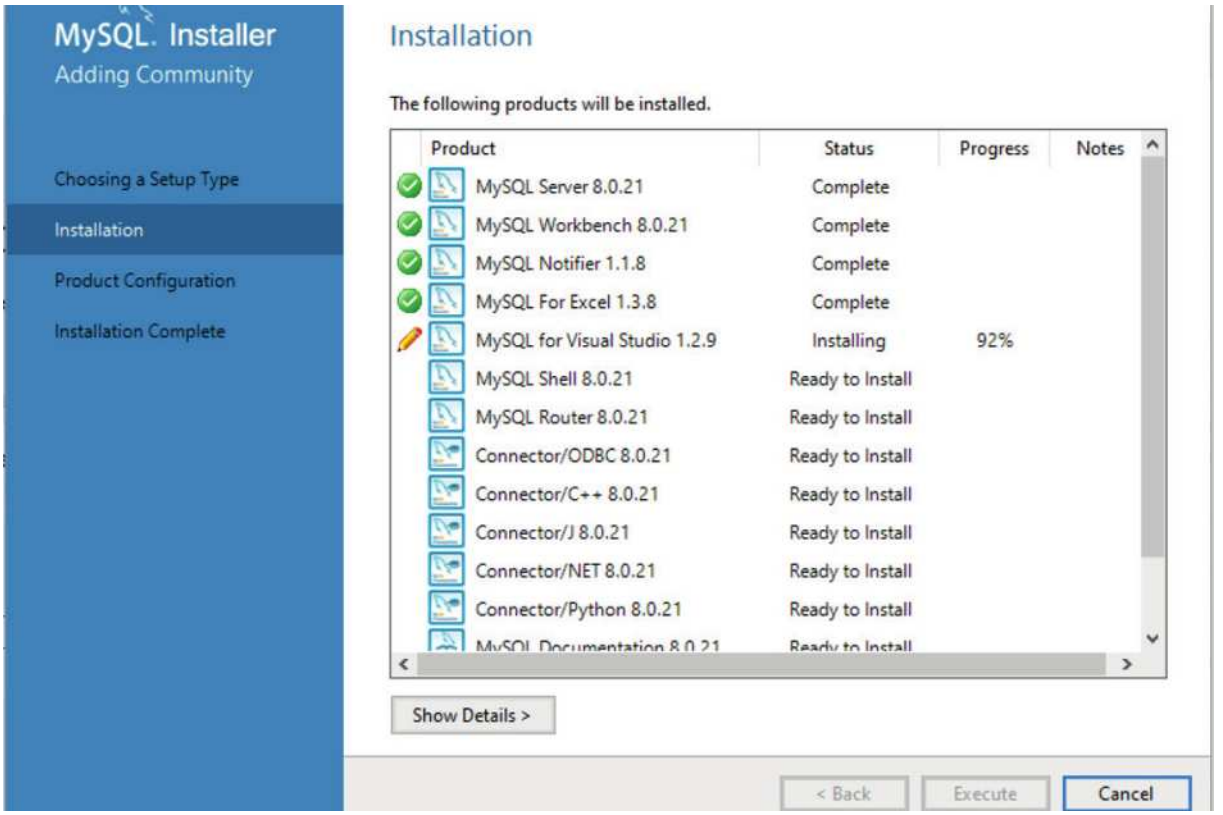


Görsel 6.7: MySQL kurulum türü seçme ekranı

Görsel 6.8'deki ekranda kurulacak bileşenler listelenmiştir. "Execute" düğmesine basılarak kuruluma başlanır (Görsel 6.9).

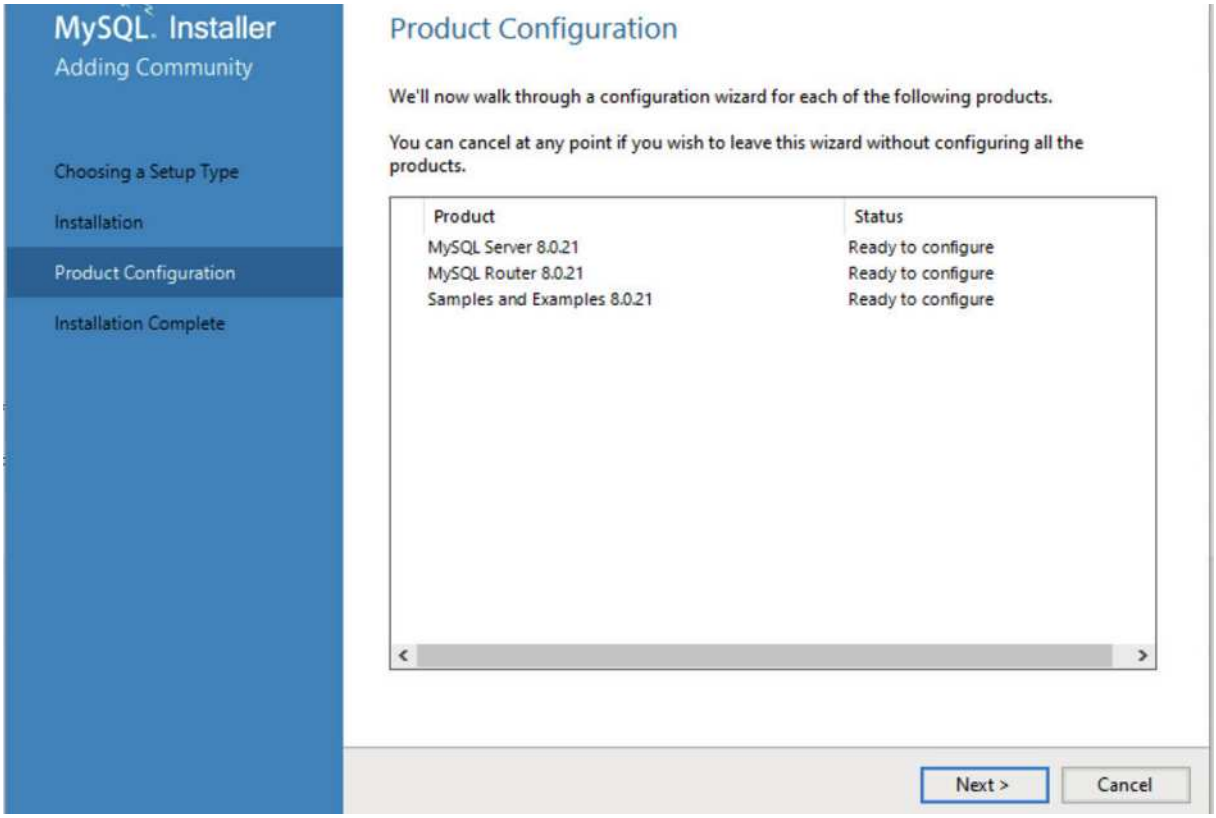


Görsel 6.8: Kurulacak bileşenler



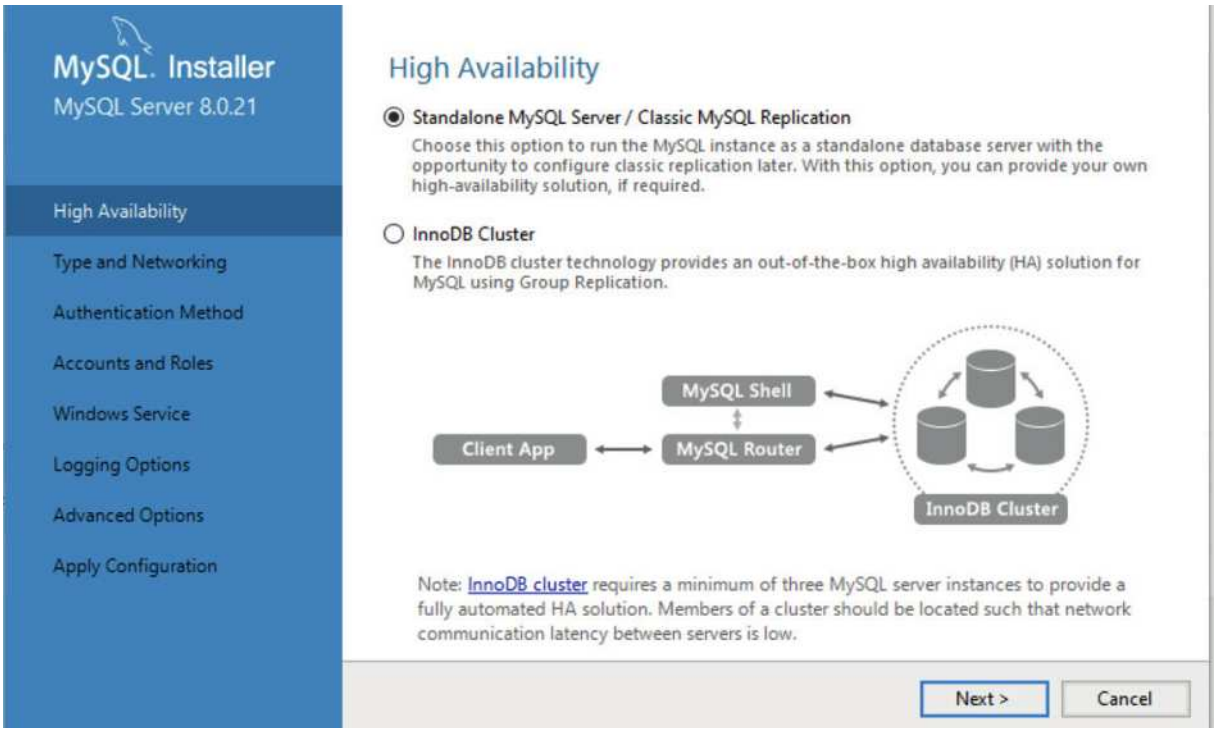
Görsel 6.9: Kurulumun başlaması

Görsel 6.10'da görüldüğü gibi kurulan bileşenler listelenir ve bu bileşenlerin ayarlanmaya hazır hâle geldiği görülür. "Next" düğmesine basılarak MySQL Server ayarlarının yapılacağı bölüme geçilir.



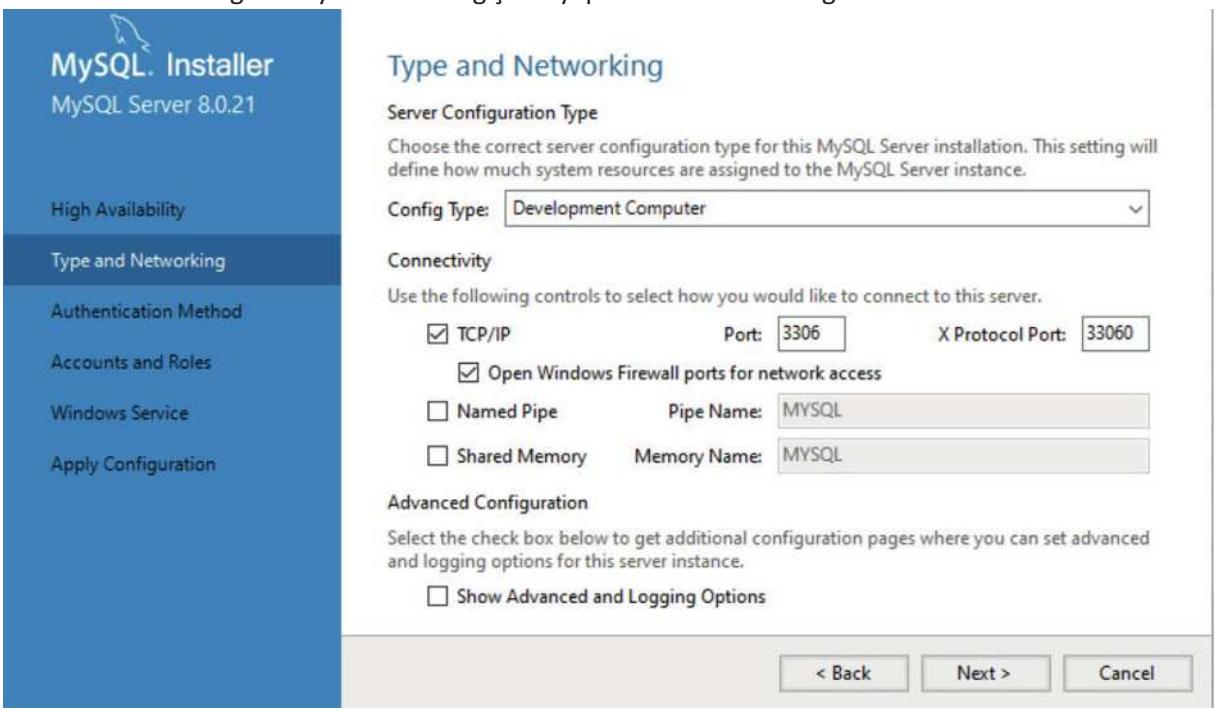
Görsel 6.10: Product Configuration ekranı

Paragrafta "Standart bir kurulum için herhangi bir ayar yapılmasına gerek yoktur. Görsel 6.11'deki ekranda "Next" düğmesine basılır.



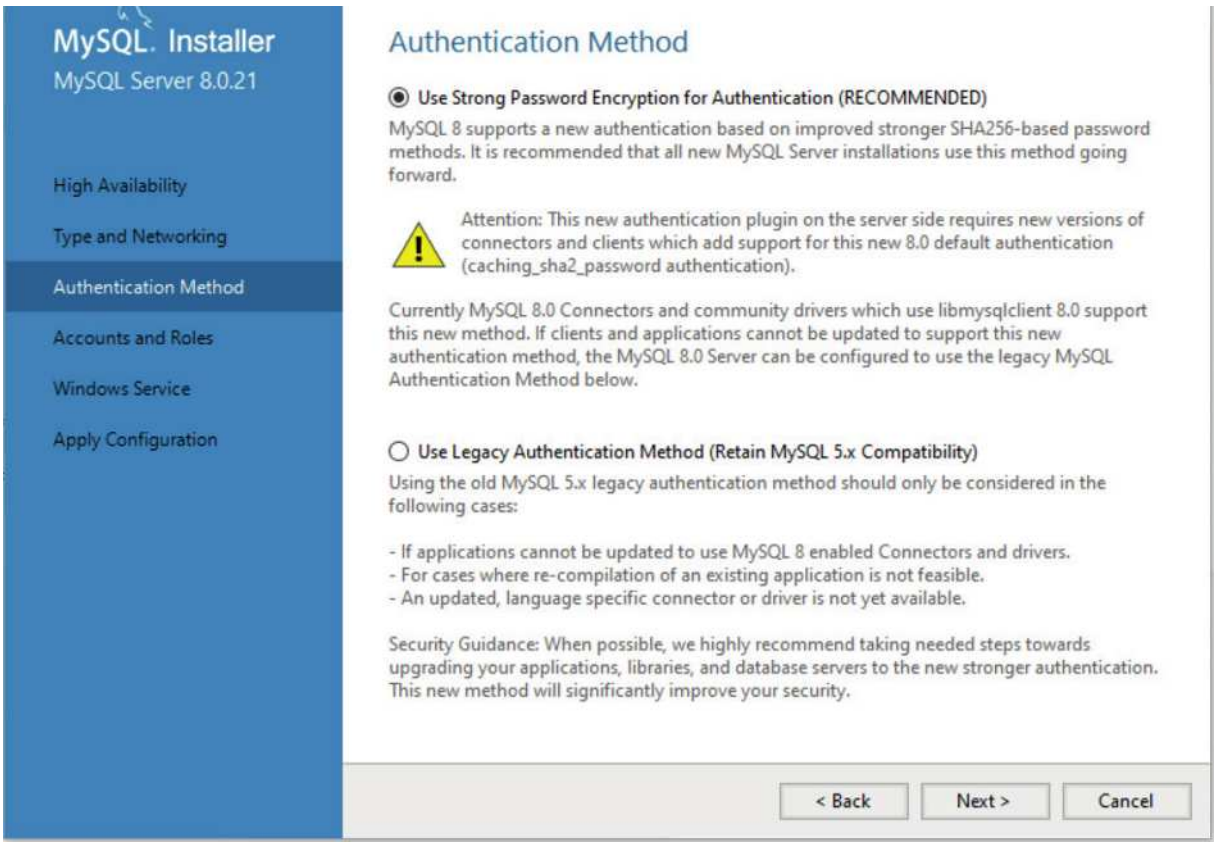
Görsel 6.11: MySQL Server ayarları

Görsel 6.12'deki bağlantı ayarlarında değişiklik yapılmadan "Next" düğmesine basılır.



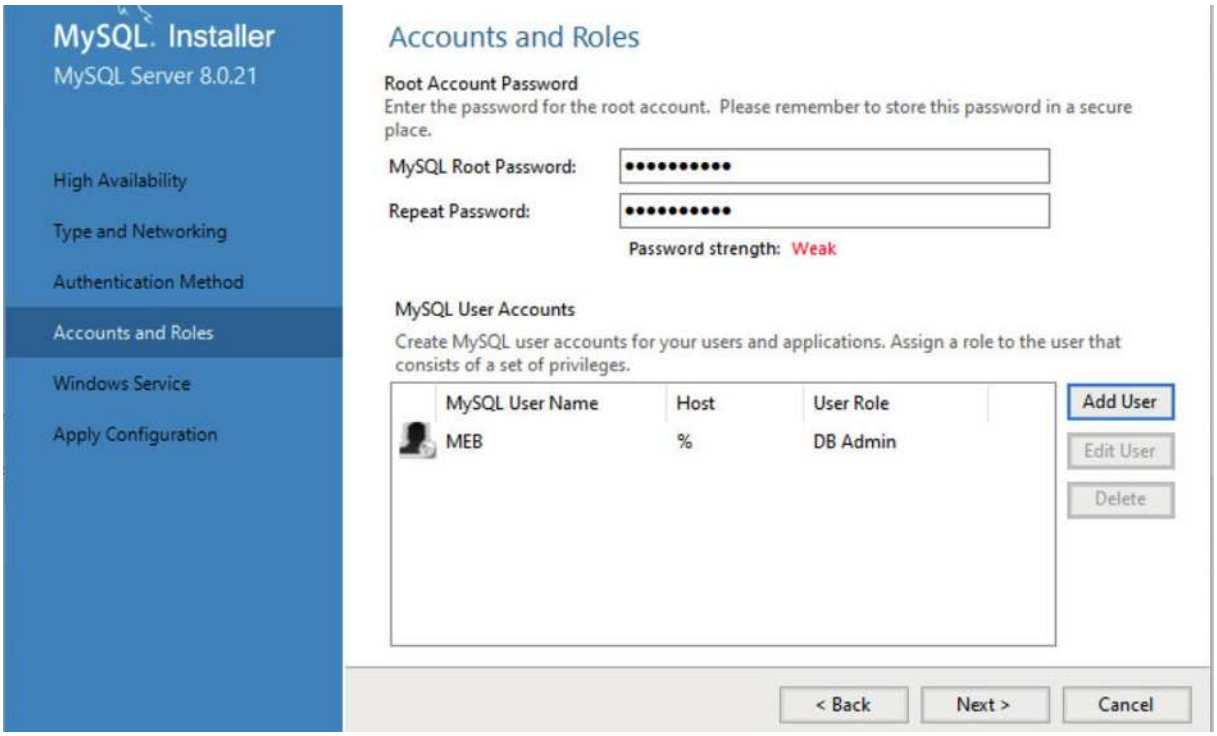
Görsel 6.12: Type and Networking ayarları

Görsel 6.13'teki ekranda MySQL Server erişim güvenliği için seçim yapılmalıdır. Bu ekranda **"Use Strong Password Encryption for Authentication"** seçeneği işaretlenerek üst düzey güvenlik ayarları seçilir.



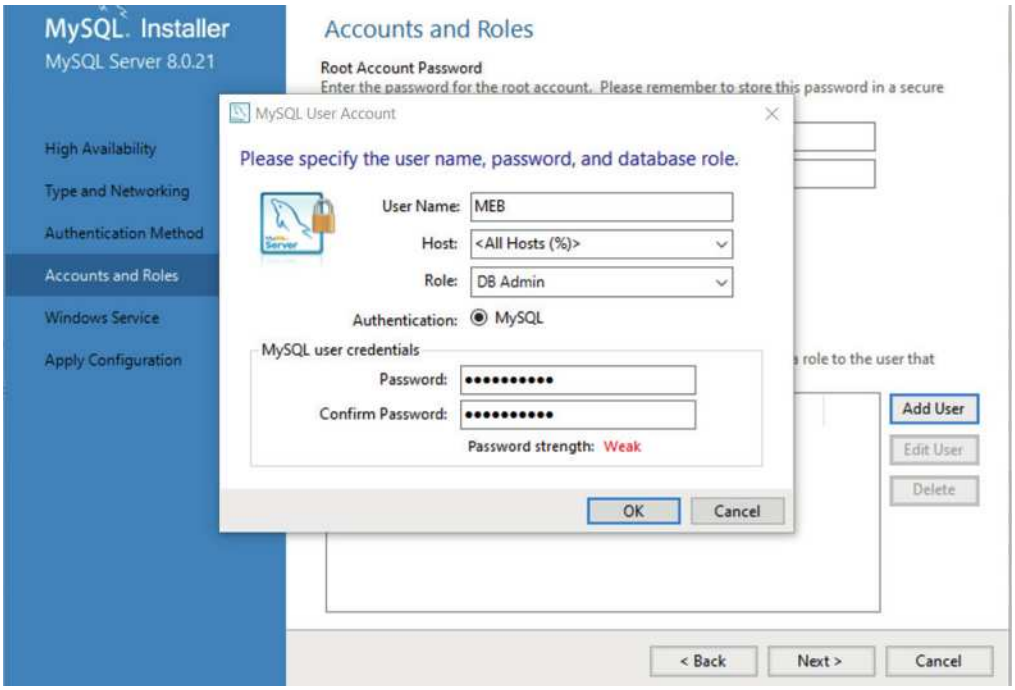
Görsel 6.13: Authentication Method ekranı

Görsel 6.14'teki ekranda MySQL Server erişimi için bir şifre belirlenmeli ve sunucuya (Server) erişecek başka kullanıcılar varsa onlar da sisteme eklenmelidir. Görsel 6.14'te üst kısımdaki varsayılan root kullanıcısı için MySQL Server şifresi belirlenmiştir. Yeni bir kullanıcı eklenmek istenirse "Add User" düğmesine basılır.



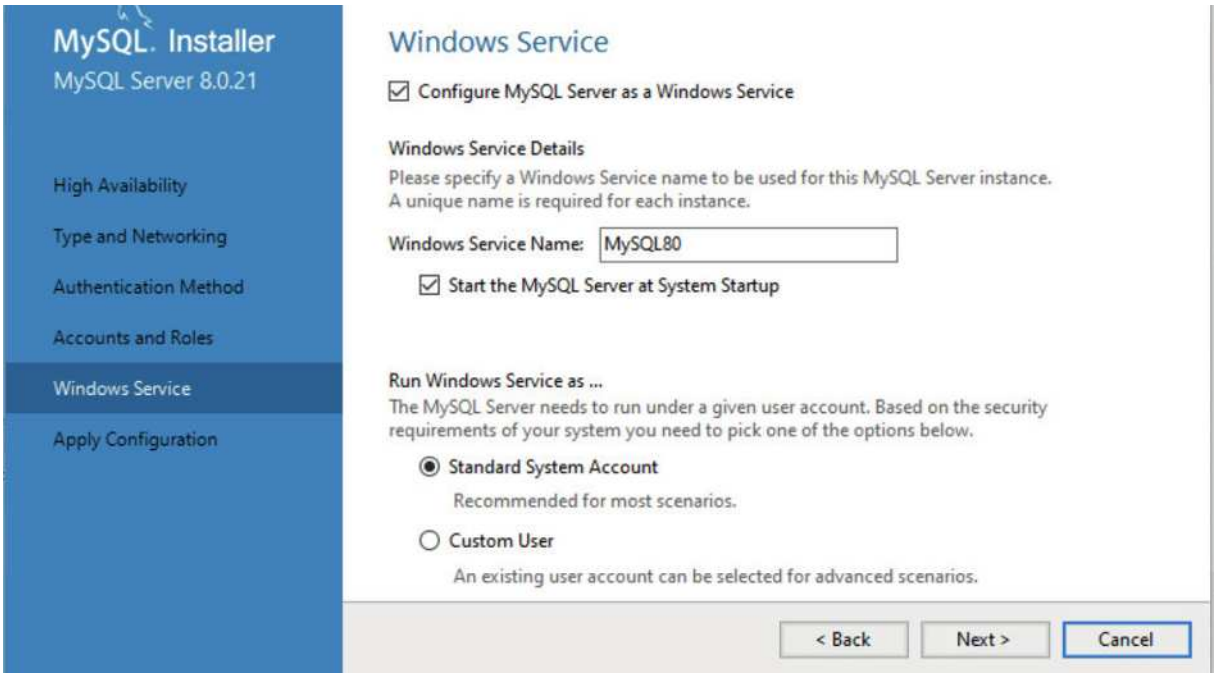
Görsel 6.14: Accounts and Roles ayarları

Görsel 6.15'teki ekranda "Add User" düğmesi tıklanarak yeni bir kullanıcı eklenebilir. Eklenen kullanıcılar "MySQL User Accounts" bölümünde görüntülenir. Kullanıcı ve şifre işlemleri bittikten sonra "Next" düğmesine basılır.



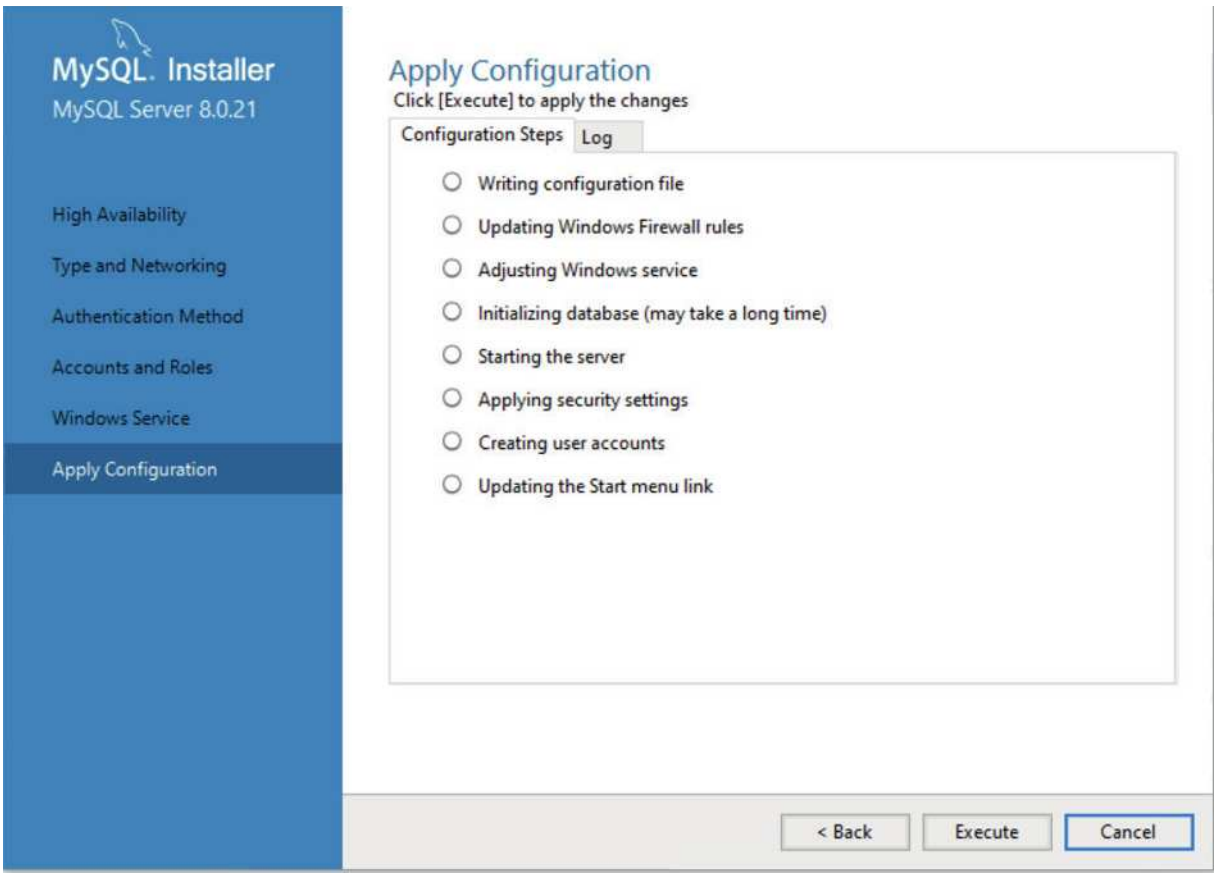
Görsel 6.15: MySQL Server'a kullanıcı eklenmesi

Görsel 6.16'daki Windows Service ayarlarında değişiklik yapılmadan "Next" düğmesine basılır.

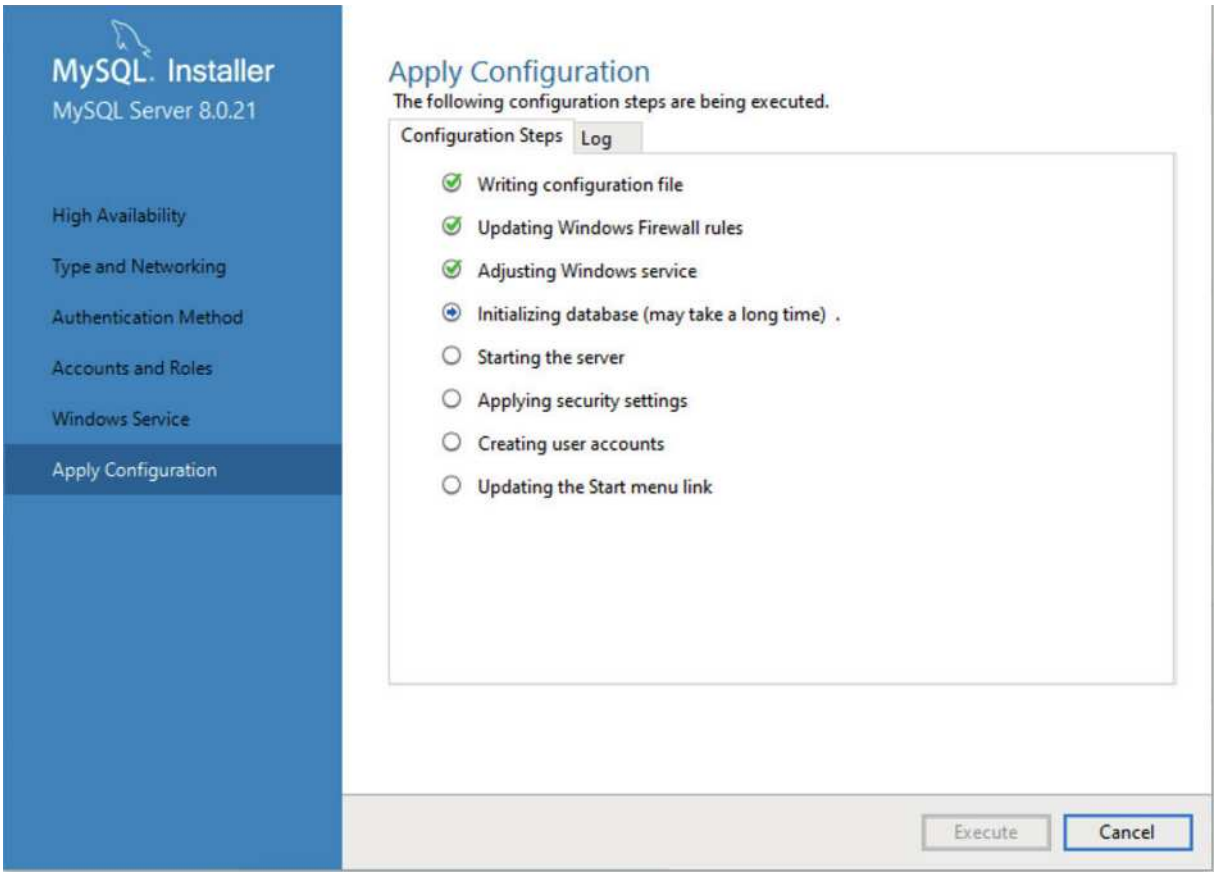


Görsel 6.16: Windows Service ayarları

Görsel 6.17'deki ekran ile MySQL Server ayarlarının uygulanması sağlanır. "Execute" düğmesine basılarak işleme devam edilir (Görsel 6.18).

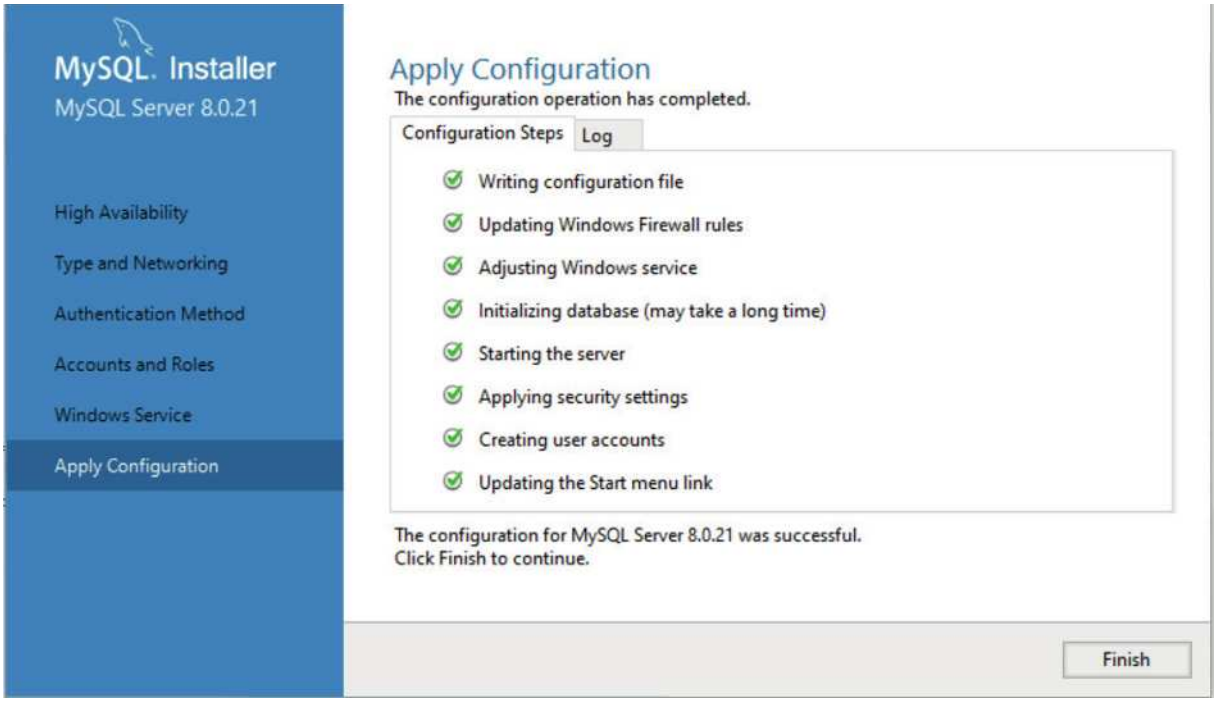


Görsel 6.17: Apply Configuration ekranı



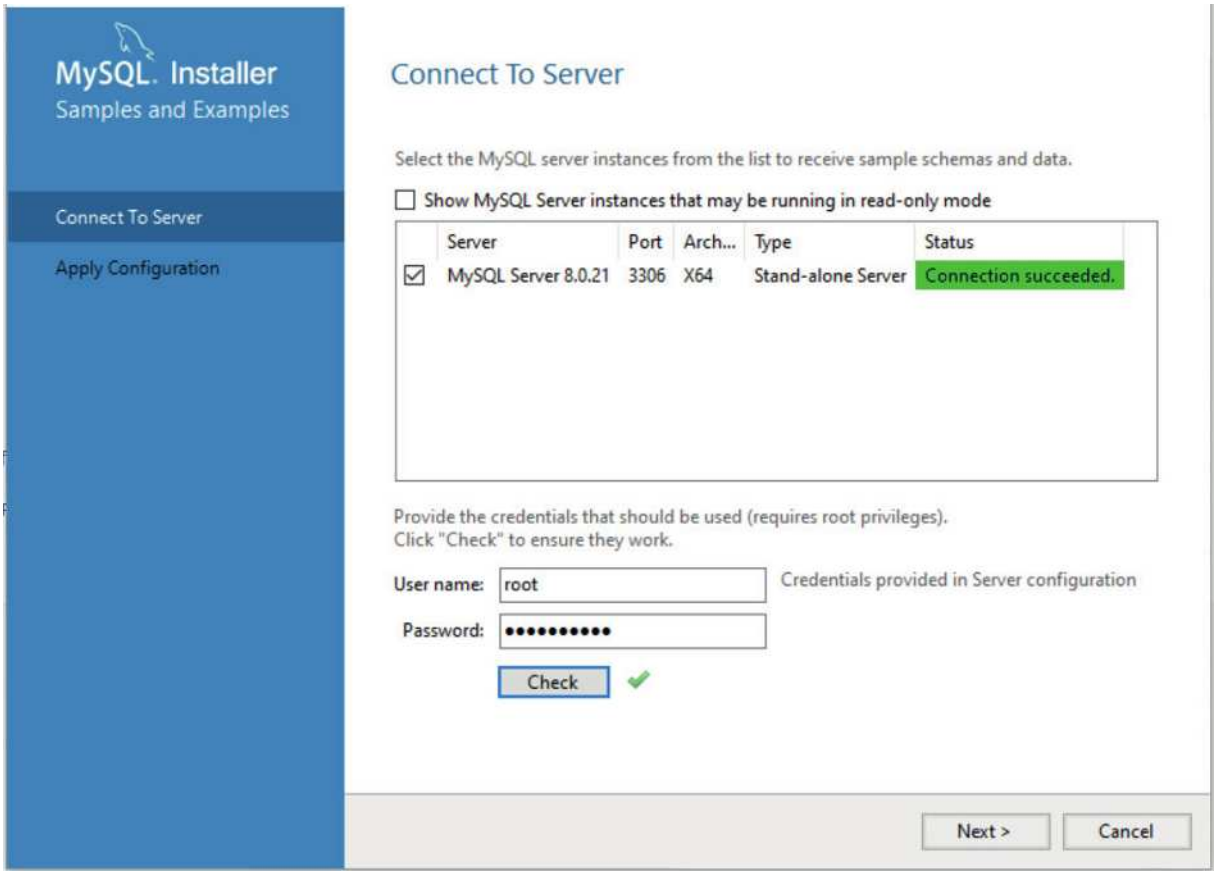
Görsel 6.18: Server ayarları uygulama ekranı

Görsel 6.19'da görüldüğü gibi bütün ayarlar doğru bir şekilde uygulanır ve MySQL Server kullanıma hazır hâle getirilir.

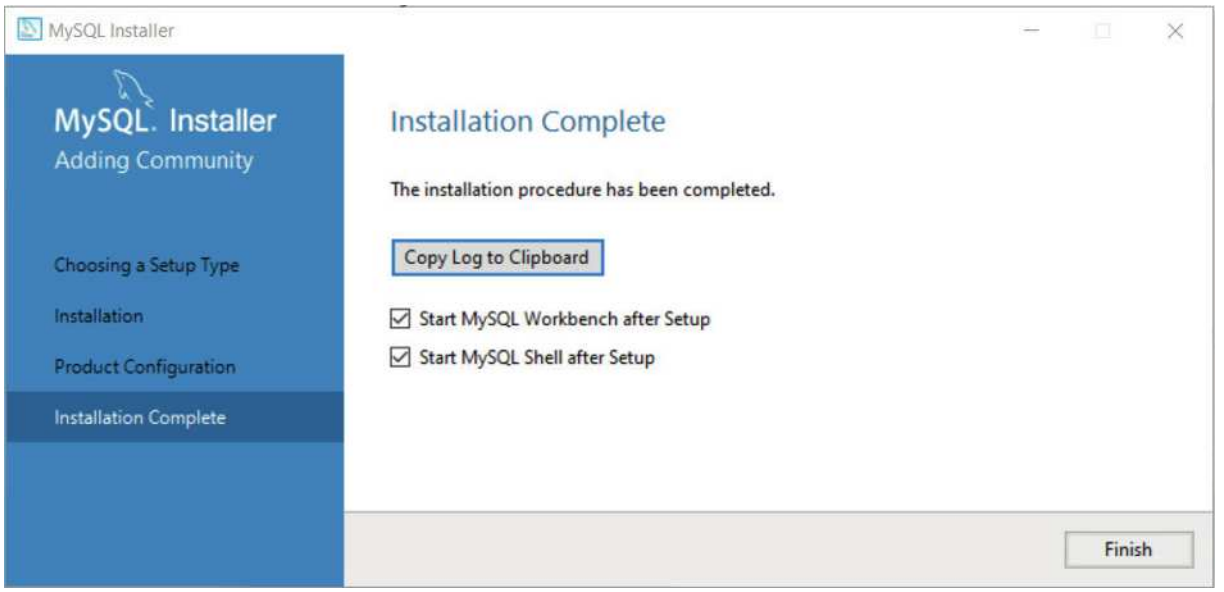


Görsel 6.19: Server ayarlarının tamamlanması

Görsel 6.20'deki ekranda kurulum aşamasında belirlenen şifre girilir ve MySQL Server'ın çalışıp çalışmadığı test edilir. "Connection succeeded" yazısı görünürse MySQL Server kurulumu ve ayarları doğru yapılmıştır. Görsel 6.21'de MySQL Server kurulumunun bitiş ekranı görülür.



Görsel 6.20: Connect To Server ekranı



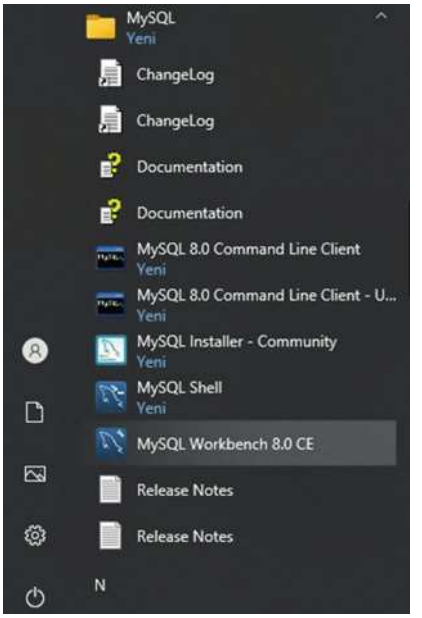
Görsel 6.21: MySQL Server kurulumunun bitiş ekranı

6.1.4. Veri Tabanı Arayüz Ekranı

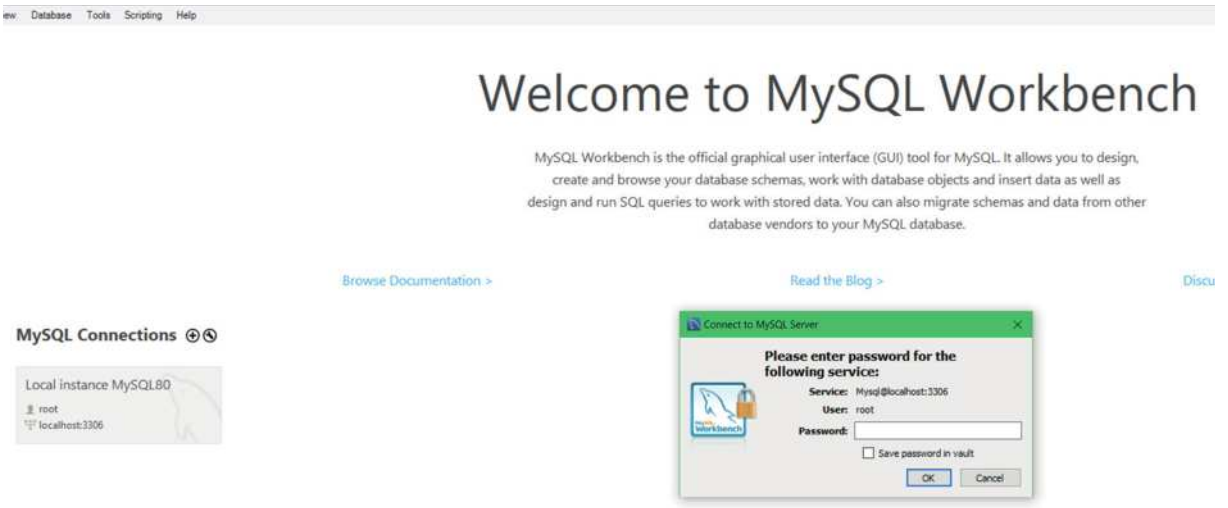
Kurulum esnasında MySQL veri tabanıyla birlikte MySQL Workbench programı da kurulur. Veri tabanı ile ilgili işlemler MySQL Workbench programı üzerinden yapılır.

MySQL Workbench programı Görsel 6.22'de görüldüğü gibi Başlat menüsünde bulunan MySQL program klasörü içinden çalıştırılır. Arama kutusuna program adı yazılarak da MySQL Workbench programı çalıştırılabilir.

Görsel 6.23'te kurulum esnasında belirlenen parola girilmelidir. Ekranın alt tarafındaki **"Save password in vault"** seçeneği işaretlenerek bir daha şifre sorulmaması sağlanabilir.

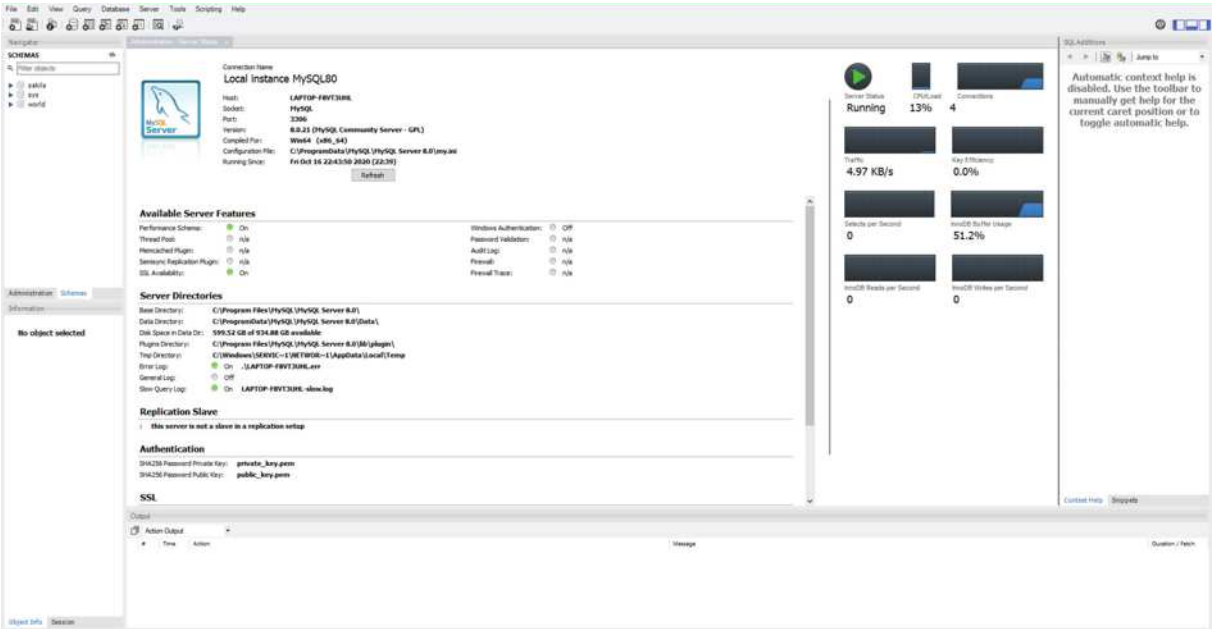


Görsel 6.22: Başlat menüsünden MySQL Workbench çalıştırma



Görsel 6.23: MySQL Workbench arayüz giriş ekranı

Şifre girildikten sonra MySQL başlangıç ekranı ile karşılaşılır. Bu ekranda MySQL Server ile ilgili bilgiler bulunur (Görsel 6.24).



Görsel 6.24: MySQL Workbench Server Status ekranı

Navigator: Bu penceredeki "Schemas" alanında veri tabanları görülür. Veri tabanı oluşturmak ve veri tabanı içinde işlem yapmak için bu bölüm kullanılır. MySQL kurulduktan sonra içinde üç adet örnek veri tabanı görülür. "Administration" bölümü ise daha çok veri tabanı ve sunucu ile ilgili yönetimsel işlemler için kullanılır.

Information: "Object Info" bölümü, seçilen bileşene ait bilgilerin (veri tabanı, tablo vb.) görülebileceği alandır. "Session" bölümü ise o anki oturum ile ilgili sunucu ve bağlantı bilgilerini içerir.

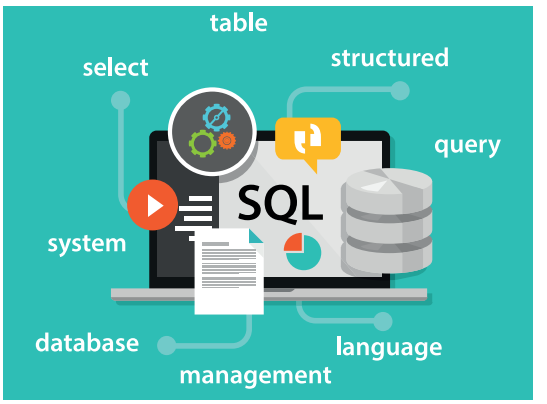
Output: Yapılan işlemlerin sonuçlarının görülebileceği alandır. İşlem doğru tamamlanmışsa yeşil renkte, işlemde hata oluşmuşsa hata mesajıyla beraber kırmızı renkte görülür.

SQL Additions: Komutlar ve bileşenlerle ilgili yardım alınabilecek alandır.

Sekmeler: MySQL veri tabanları ile ilgili bütün işlemlerin yapıldığı alandır. Orta alanın tamamını kullanır. Erişilen bütün bileşenlerle ilgili işlemler bu alanda açılacak sekmelerin içinde yapılır.

6.1.5. SQL (Structured Query Language)

Structured Query Language (Yapılandırılmış Sorgu Dili); veriyi sorgulamak, manipüle etmek, tanımlamak ve veriye erişim kontrolü sağlamak üzere veri tabanlarında kullanılan bir dildir (Görsel 6.25). Veri tabanı ile ilgili yönetimsel işlemler de SQL dili ile yapılabilir. SQL günümüzde oldukça yaygın bir şekilde kullanılsa da SQL dilinin daha etkili kullanılabilmesi için bazı firmalarca PL-SQL, T-SQL dilleri geliştirilmiştir.



Görsel 6.25: SQL (Structured Query Language)

6.2. VERİ TABANI TASARIMI

Veri tabanı tasarımı şu adımlara göre yapılır:

Veri Tabanının Amacını Belirleme

Öncelikle veri tabanının hangi verileri saklamak için tasarlanacağı belirlenir. Örneğin telefon rehberi, kütüphane programı, stok takibi, öğrenci bilgi sistemi, ticaret, blog sitesi, teknik servis programı vb.

Gerekli Bilgileri Bulma ve Düzenleme

Ürün adı, sipariş numarası, müşteri adı, tedarikçi, stok bilgisi gibi veri tabanına kaydedilmesi istenen tüm bilgi türleri toplanır. Bu bilgilerin toplanabilmesi için veri tabanının amacı belirlenmelidir.

Bilgileri Tablolara Bölme

Bilgi ögeleri ürünler, siparişler ve müşteriler gibi ana birimlere veya konulara bölünür. Veri tabanında her nesne için bir tablo oluşturulur. Bir başka deyişle her tabloyla yalnızca bir nesne temsil edilir.

Bilgi Ögelerini Sütunlara Dönüştürme

Her tabloda hangi bilgilerin depolanacağına karar verilir. Her öge, bir alana dönüşür ve tabloda bir sütun olarak görüntülenir. Örneğin ürünler tablosunda ürün adı, ürünün kategorisi gibi alanlar veya müşteriler tablosunda müşteri adı, e-posta adresi gibi alanlar depolanır.

Birincil Anahtarları Belirtme

Her tablonun birincil anahtarı seçilir. Birincil anahtar, her satırı benzersiz olarak tanımlamak için kullanılan bir sütundur. Örneğin kişi bilgilerinin tutulacağı bir tabloda T.C. Kimlik Numarası birincil anahtar olarak seçilebilir çünkü bu değerle yalnızca bir kişi temsil edilir. Ürünler için bir tablo oluşturulduğu varsayılırsa her ürünün benzersiz bir ürün kodu değeri birincil anahtar olarak belirtilebilir.

Tablo İlişkilerini Ayarlama

Her tablo kontrol edilerek bu tablodaki verilerin diğer tablolardaki verilerle ilişkisine karar verilir. Gerekirse ilişkileri netleştirmek için tablolara alanlar eklenir veya yeni tablolar oluşturulur.

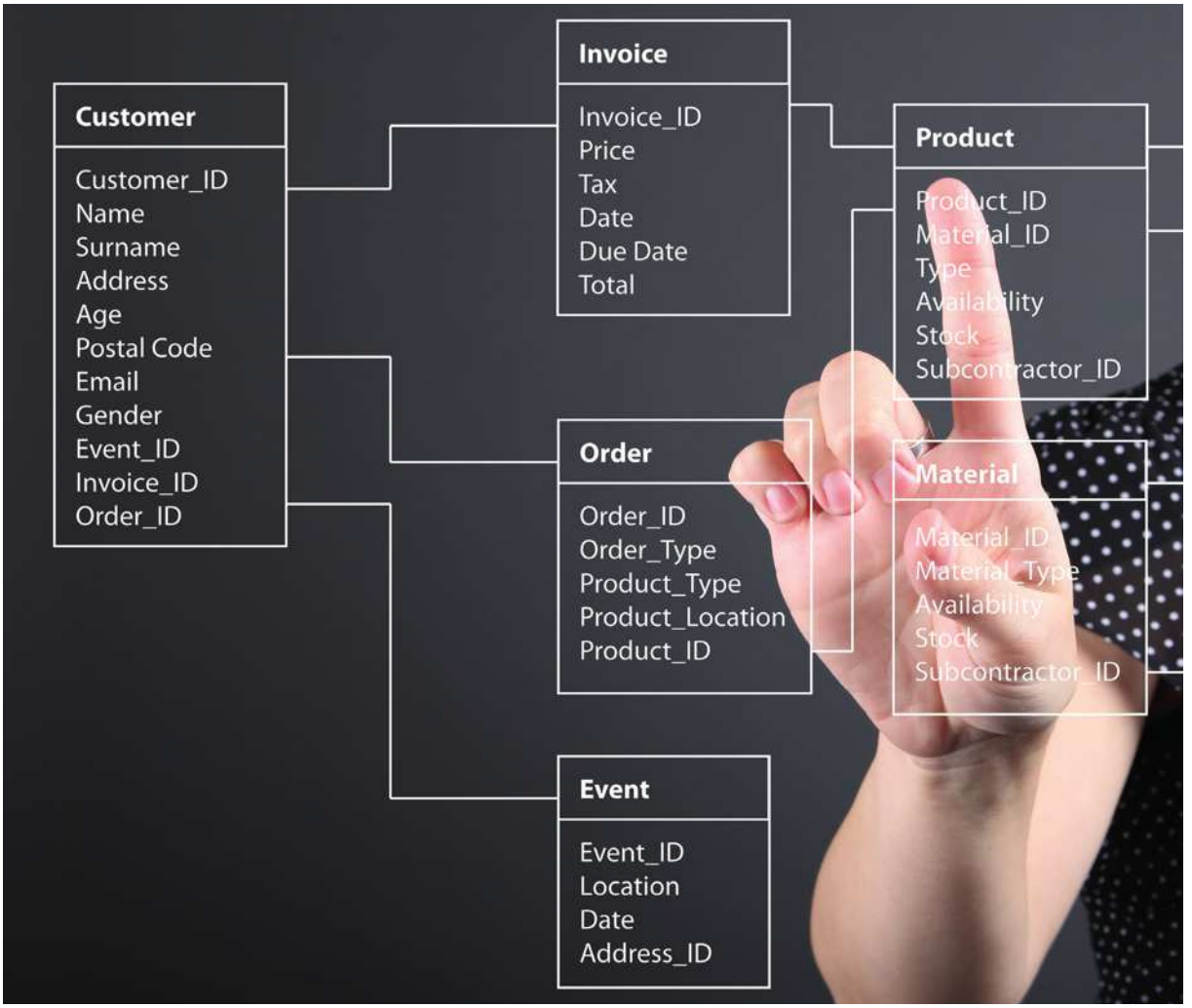
Tasarımı İyileştirme

Tasarım, hatalar açısından çözümlenir. Tablolar oluşturulur ve örnek veriler için birkaç kayıt eklenir. Tablolardan istenen sonuçların alınıp alınamadığına bakılır. Tasarımda gereken ayarlamalar yapılır.

Normalleştirme Kurallarını Uygulama

Tabloların doğru yapılandırılıp yapılandırılmadığını görmek için veri normalleştirme (normalizasyon) kuralları uygulanır. Tablolarda gereken ayarlamalar yapılır.

Görsel 6.26'da örnek bir veri tabanı tasarımı gösterilmiştir.



Görsel 6.26: Veri tabanı tasarımı

6.2.1. Normalizasyon

Normalizasyon (Ayrıştırma), veri tabanlarında çok fazla sütun ve satırdan oluşan bir tabloyu tekrarlardan arındırmak için daha az satır ve sütun içeren alt kümelere ayırma işlemidir. Normalizasyon aynı zamanda taslak şeklinde hazırlanan veri tabanının revizyonlar yapılarak son hâline getirilmesini sağlayan bir yöntemdir.

Veri tabanı içinde oluşturulan tabloların mantıksal olarak ve performans açısından iyi tasarlanması gerekir. İlişkisel veri tabanı tasarlanırken veri tekrarını, veri kaybını veya veri yetersizliğini önlemek için normalizasyon işlemi uygulanır.

Normalizasyonun Amaçları

- Veri bütünlüğünü ve tutarlılığını sağlamak
- Veri tekrarını ortadan kaldırarak veri tabanı boyutunu azaltmak
- Performansı artırmak
- Uygulamadan bağımsızlık (Veri tabanı, yapılacak uygulamaya göre değil, kaydedilecek verilere göre

tasarlanmalıdır. Bu durumda veri tabanı her uygulamada kolaylıkla kullanılabilir.)

- Ekleme, silme, güncelleme ve listeleme işlemlerinde ortaya çıkabilecek aksaklıkları önlemek

Normalizasyonun Aşamaları

Normalizasyon beş aşamadan oluşur. Bu aşamalar şunlardır:

- Birinci Normal Form
- İkinci Normal Form
- Üçüncü Normal Form
- Boyce-Codd Normal Formu
- Dördüncü Normal Form

İlk üç formun uygulandığı veri tabanı, normalizasyon kurallarına uygun olarak kabul edilir.

Normal Olmayan Form

Normal olmayan form, normalizasyon kuralları uygulanmamış bir veri tabanı tasarımı anlamına gelir. Görsel 6.27'de verilen tablo, bir kargo firmasına ait verilerin kaydedildiği normal olmayan bir forma aittir. Kargo firmasına ait verilerin kaydedildiği bu veri tabanı normalizasyon kurallarına uygun hâle getirilmelidir.

müşteri_no	şehir_kodu	şehir_adı	gönderi_no	miktar
1	34	İstanbul	1,2,3,4,6	300,200,400,200,100
2	6	Ankara	1,2	300,400
3	6	Ankara	2	200
4	34	İstanbul	2,4,5	200,300,400

Görsel 6.27: Normal olmayan form örneği

1. Birinci Normal Form (1NF)

Birinci Normal Form kuralları şunlardır:

- Aynı tablo içinde tekrarlayan kolonlar bulunamaz.
- Her kolonda yalnızca bir değer bulunabilir.
- Her satır, bir eşsiz anahtarla tanımlanmalıdır (Unique Key - Primary Key).

İlişkisel veri tabanı modelinin temel kuralına göre bütün niteliklerin (sütunların) aldığı değerler tek ve basit olmalıdır. Görsel 6.27'deki tablo örneğinde buna uyulmadığı görülür. Görsel 6.27'de verilen örnek tablo incelendiğinde gönderi_no ve miktar alanları aynı anda birden fazla değer almıştır. Bu alanlar, her bir alanda tek değer olacak şekilde düzenlenmelidir.

Düzenleme işlemi yapıldığında Görsel 6.28'deki tablo elde edilir.

müşteri_no	şehir_kodu	şehir_adı	gönderi_no	miktar
1	34	İstanbul	1	300
1	34	İstanbul	2	200
1	34	İstanbul	3	400
1	34	İstanbul	4	200
1	34	İstanbul	6	100
2	6	Ankara	1	300
2	6	Ankara	2	400
3	6	Ankara	2	200
4	34	İstanbul	2	200
4	34	İstanbul	4	300
4	34	İstanbul	5	400

Görsel 6.28: Birinci Normal Form uygulanan tablo

Görsel 6.28'deki tablo 1NF'ye uygun hâle getirilse de ekleme, silme ve güncelleme sorunlarının devam ettiği görülür.

2. İkinci Normal Form (2NF)

İkinci Normal Formda benzersiz bir alana diğer sütunların tam bağımlı olması gerekir.

Görsel 6.28'deki 1NF uygulanan tablo incelenirse şehir_kodu ve şehir_adı alanlarının tekrarlandığı görülür. Tekrarlanan veriler; ekleme, silme, güncelleme işlemlerinde sorunlar yaratır. Sorunların giderilmesi için 2NF uygulanmalıdır. Bu uygulama ile tek tablo hâlinde tutulan bilgiler "Müşteriler" ve "Miktar" olarak iki tabloya ayrılır (Görsel 6.29).

Müşteriler Tablosu

müşteri_no	şehir_kodu	şehir_adı
1	34	İstanbul
2	6	Ankara
3	6	Ankara
4	34	İstanbul

Miktar Tablosu

müşteri_no	gönderi_no	miktar
1	1	300
1	2	200
1	3	400
1	4	200
1	6	100
2	1	300
2	2	400
3	2	200
4	2	200
4	4	300
4	5	400

Görsel 6.29: İkinci Normal Form uygulanması

Görsel 6.29'da 2NF ile güncelleme sorununun ortadan kalktığı görülür fakat ekleme ve silme sorunları devam eder. Verilen tablolara yeni bir şehir eklemek için yeni bir müşterinin de eklenmesi gerekir. Ayrıca bu tablodan kayıt silme işlemi yapılmak istenirse ve bu şehirde kayıtlı başka bir müşteri yoksa şehir

bilgisi de silinir. Bu iki sorunu ortadan kaldırmak için Üçüncü Normal Form uygulanması gerekir.

3. Üçüncü Normal Form (3NF)

Üçüncü Normal Form uygulanırken her kolon, benzersiz anahtara tam bağımlı olmalıdır (Anahtar olmayan başka bir kolona bağımlı olmamalıdır.). Ayrıca 3NF'de veri tekrarını azaltmak için tanım tabloları oluşturulur. Görsel 6.30'da ekleme ve silme sorunlarının ortadan kaldırıldığı görülür. Müşteri ve şehir bilgileri ayrı ayrı eklenebilir ve silinebilir.

Şehirler Tablosu

şehir_kodu	şehir_adı
6	Ankara
34	İstanbul
35	İzmir

Müşteriler Tablosu

müşteri_no	şehir_kodu
1	34
2	6
3	6
4	34
5	35

Miktar Tablosu

müşteri_no	gönderi_no	miktar
1	1	300
1	2	200
1	3	400
1	4	200
1	6	100
2	1	300
2	2	400
3	2	200
4	2	200
4	4	300
4	5	400

Görsel 6.30: Üçüncü Normal Form uygulanması

6.2.2. Veri Türleri

C# programlama dilinde değişkenlerde saklanacak bilgiye göre veri tipi belirlendiği gibi veri tabanı tasarımı yapılırken de her kolon (alan) için veri tipi tanımlanmalıdır. Bu tanımlamalar yapılırken dikkat edilmesi gereken noktalardan biri, mümkün olduğunca en uygun ve en az yer kaplayacak veri tipinin seçilmesidir (Tablo 6.1).

Tablo 6.1: Veri Türleri

Veri Tipi	Türü	Açıklama
char(n)	String	Uzunluğu değişmeyen sabit verileri saklar. n değeri 10 ise daha kısa değer girilince kalan boşluğu kendi tamamlar ve öylece saklar.
varchar(n)	String	Değişebilir uzunluktaki verileri saklar. En fazla 8.000 karakter alır. n değeri maksimum değerdir. Daha kısa değer girilse bile olduğu gibi kaydeder.
text	String	Değişebilir uzunluktaki karakterleri saklar. En fazla 2 GB metin içerir.
nchar	String	Sabit uzunluktaki Unicode karakterleri saklar. Char tipinden farkı, çoklu dil ve Unicode desteği olmasıdır. En fazla 4.000 karakterlik değer tutar.
nvarchar	String	Değişebilir uzunluktaki verileri saklar. varchar tipinden farkı, çoklu dil ve Unicode desteği olmasıdır. En fazla 4.000 karakterlik değer tutar.
bit	Boolean	64 bit uzunluğunda binary (0 ve 1) tipinde verileri saklar.
tinyint	Tam sayı	0 ile 255 arasında değerleri saklar.
smallint	Tam sayı	-32.768 ile 32.767 arasında değerleri saklar.
int	Tam sayı	-2.147.483.648 ile 2.147.483.647 arasında değerleri saklar.

Tablo 6.1'in devamı sonraki sayfada

Veri Tipi	Türü	Açıklama
bigint	Tam sayı	-9.223.372.036.854.775.808 ile 9.223.372.036. 854.775.807 arasında değerleri saklar.
datetime	Tarih	1 Ocak 1753-31 Aralık 9999 aralığında tarih değeri saklar. 3,33 milisaniye doğruluk hassasiyeti vardır.
smalldatetime	Tarih	1 Ocak 1900-6 Haziran 2079 aralığında tarih değeri saklar. 1 dakikalık doğruluk hassasiyeti vardır.
date	Tarih	1 Ocak 0001-31 Aralık 9999 aralığında tarih değeri saklar. Sadece tarih içerir, saati saklamaz.
decimal	Ondalık	Decimal veri türü hafızada ondalık M,D türünden veri tutmak için kullanılır. M değeri, virgülden önceki ve sonraki toplam basamak sayısını verir. D değeri ise virgülden sonraki basamak sayısını verir. Örneğin 102,35 sayısını hafızada tutmak için DECIMAL(5,2) şeklinde veri türü tanımlanmalıdır. Buradaki 5 değeri virgülsüz toplam basamak değerini, 2 değeri ise virgülden sonraki basamak değerini temsil eder.
float	Ondalık	Float veri türü, ondalık sayıların depolanabileceği bir veri türüdür. Bu veri türüne sahip sütun tanımlanırken sayının tam ve ondalık kısmının basamak değeri belirtilir.

Bunlar dışında da veri türleri mevcuttur.

6.2.3. Veri Tabanı Oluşturma

VTYS programlarında veri tabanı oluşturmak için SQL komutları veya program arayüzü kullanılır. SQL sorgusu ile veri tabanının oluşturulması için sorgu ekranında Create Database "Veritabanı Adı" komutu çalıştırılır.

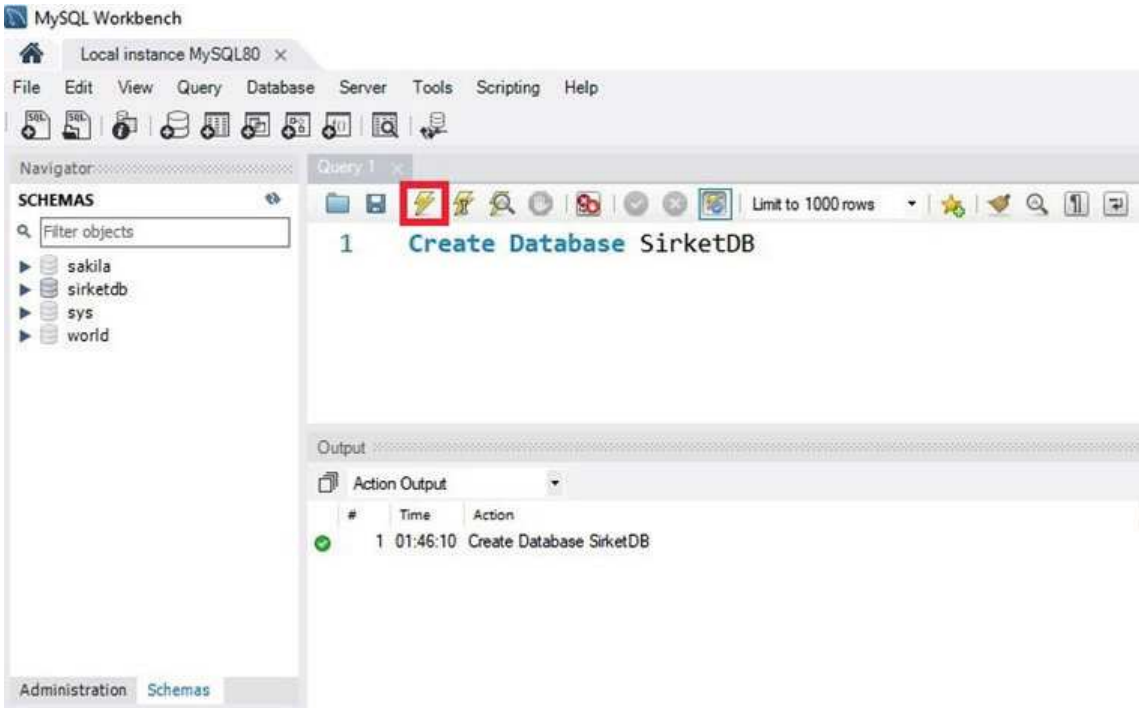
Not : Veri tabanı tasarımında veri tabanı adı, tablo adı, alan adları gibi nesneler isimlendirilirken o nesne ile ilişkili bir isim kullanılması önemlidir. Ayrıca bu isimlendirmelerde Türkçe karakter kullanılmamasına özen gösterilmelidir ve değişken isimlendirme kurallarına uyulmalıdır.

Bu bölümde bir şirkette çalışan personel bilgilerinin tutulacağı veri tabanı oluşturulacaktır. Bu yüzden veri tabanına **"SirketDB"** ismi verilecektir. Veri tabanı isimlendirilirken dosya isminin yanında VT (Veri tabanı) veya DB (Database) gibi veri tabanı olduğunu belirten ifadeler konulabilir.

MySQL Workbench ekranında sorgu yazılması için öncelikle üst bölümdeki (SQL sorgusu ekleme) simgesine tıklanır.

Veri tabanı oluşturma komutu yazıldıktan sonra üst bölümdeki simgesine tıklanarak sorgu çalıştırıldığında "Output" bölümünde işlemin başarılı olduğunu belirten bir mesaj gösterilir ve "SirketDB" isimli veri tabanı oluşturulur.

SCHEMAS menüsünün yanındaki yenileme simgesine tıklanarak veri tabanının o bölümde görünmesi sağlanır (Görsel 6.31).



Görsel 6.31: Veri tabanı oluşturma

Oluşturulan bir veri tabanının silinmesi için üzerine sağ tıklanarak **Drop Schema** seçeneği seçilir. Ayrıca "Drop Database VeriTabanıAdı" şeklindeki komut yapısıyla da silme işlemi yapılabilir. Oluşturulan veri tabanı üzerinde değişiklik yapılması için üzerine sağ tıklanarak **Alter Schema** seçeneği seçilir.

Sıra Sizde

MySQL sunucunuzda Deneme_VT isimli bir veri tabanı oluşturunuz.

6.2.4. Veri Tabanında Anahtarlar (Keys) ve İndeksler

Primary Key (PK-Birincil Anahtar)

Primary Key, bir veri tabanı tablosundaki her satır için bir vekil veya tanımlayıcı **(identify)** görevi görür, kısıtlamadır **(constraint)** ve eşsizdir **(unique)**. Bir başka deyişle her PK yalnızca o satırdaki verileri temsil etmelidir. Satırlara ait değerlerin karışmaması adına bu alana ait bilgi tekrarlanamaz, boş geçilemez ve Null değeri alamaz. Birincil anahtarın temel işlevi, verilerin hangi satıra dizileceğine veya hangi satırda düzenleneceğine karar vermesidir. Çoğunlukla tek bir alan olarak kullanılsa da birden fazla alanın birleşimiyle de oluşturulabilir. Sayılar genelde birincil anahtar olarak seçilir ancak bu bir zorunluluk değildir. **İlişkisel veri tabanlarında** (Relational Database) mutlaka birincil anahtar olmalıdır.

Primary Key; Id, tcKNo (Tc kimlik no), kullaniciAdi (kullanıcı adı) gibi yalnızca bir kullanıcıya ait olabilecek alanlardan seçilmelidir. Ad, soyad, bölüm, doğum yeri gibi birden fazla kullanıcıyı temsil edebilecek değerler PK olamaz. PK genelde otomatik artan sayı (auto increment) olarak kullanılır. Bu alana "id" denilir ve değer girilmez. Kullanıcı diğer alanlara bilgi girişi yaptığında bu alanda bulunan bir önceki id değeri otomatik artarak yazılır.

Foreign Key (FK-Yabancı Anahtar)

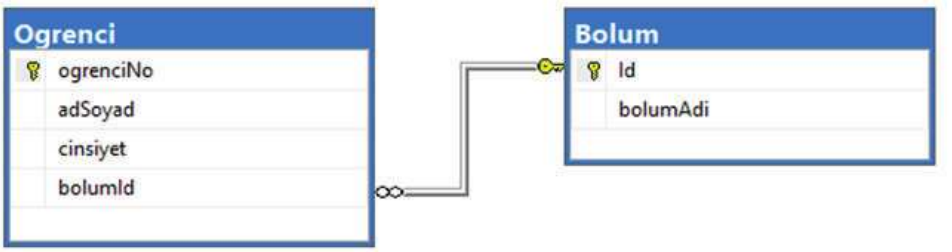
Foreign Key, ikincil anahtar olarak da ifade edilir. Bir veri tablosuna girilebilecek değerleri başka bir veri tablosundaki alanlarla ilişkilendirmeye yarar. Foreign Key, bir tablonun birincil anahtar değerinin bir başka tablo içinde yer almasıdır. Çoğunlukla bir ana tablo (parent) ile alt tablonun (child) ilişkilendirilmesinde kullanılır. Bu sayede olası veri tekrarlarının önüne geçilir ve kullanıcının yanlış veri girişi yapması önlenir.

Unique Key (UQ-Tekil veya Benzersiz Anahtar)

Unique Key, bulunduğu tablo içinde, bir değeri sadece bir kere alır. İlgili değerin tekrar atanmasına izin verilmez. Bu anahtarın işlevi birincil anahtar ile benzerlik taşısa da ikisi arasındaki en önemli farklılık, Unique Key'in Null değeri alabilmesidir.

Not Her **"Primary Key"**, özellik olarak bir **"Unique Key"** olsa da tersi için aynı durum geçerli değildir.

Bir tabloda benzersiz değer alabilecek bir alan varsa ve bu alan PK değilse UQ olarak ayarlanabilir.



Ogrenci

ogrenciNo	adSoyad	cinsiyet	bolumId
197	Murat Ş.	Erkek	3
266	Elif B.	Kız	5
302	Muhammer D.	Erkek	1
456	Ümit Y.	Erkek	4
533	Salih K.	Erkek	4
557	Şule Y.	Kız	1
684	Esra S.	Kız	1
821	Mehmet Ç.	Erkek	3
845	Aysel A.	Kız	1
942	Turan G.	Erkek	4
981	Fatma Y.	Kız	5

Bolum

Id	bolumAdi
1	Bilişim Teknolojileri
2	İnşaat Teknolojisi
3	Tesisat Teknolojisi ve İklimlendirme
4	Elektrik-Elektronik Teknolojisi
5	Çocuk Gelişimi ve Eğitimi

Görsel 6.32: Primary Key ve Foreign Key kullanımı

Görsel 6.32 incelendiğinde "Ogrenci" tablosunun PK alanı **ogrenciNo** ve "Bolum" tablosunun PK alanı ise **id** olarak belirlenmiştir. Ayrıca Ogrenci tablosundaki **bolumid** alanı Bolum tablosundaki **id** alanı ile ilişkilendirilerek FK olarak ayarlanmıştır. Bu ilişki oluşturulduktan sonra Ogrenci tablosundaki bolumId alanına bolum tablosundaki id değerlerinin dışında bir bölüm girişi yapılması engellenmiştir. Bu sayede bölüm adının sürekli olarak tekrar etmesinin ve aynı zamanda kullanıcıların yanlış bölüm adı değeri girmesinin önüne geçilmiştir.

Veri Tabanında İndeks Kullanımı

İndeks, tablolarda bulunan kolonlardaki verilerin belirli bir düzene göre sıralanmasıdır. Bu sıralamanın amacı, istenen veriye çok daha hızlı sürede erişilmesidir. Özellikle büyük boyutlu veri tabanlarında çok fazla sorgulama yapılan alanlar indeks şeklinde belirtilerek listeleme sorgularının hızlı çalışması sağlanır çünkü veri tabanı ile sorgulama işlemi, indekslenen alanlara göre yapılır. Küçük boyutlu veri tabanlarında indeksleme işlemine gerek yoktur.

Primary Key şeklinde belirlenen alanların indeks olarak ayarlanmasına gerek yoktur. Bu alanlar otomatik olarak indekslenir.

6.3. TABLO İŞLEMLERİ

Tablolar, veri tabanında bilgilerin saklandığı nesnelerdir. Tablolar, alanlardan oluşur. Her bir özellik, tablo içindeki alanlara (sütunlara) karşılık gelir. Veriler, tabloların içinde satırlar hâlinde kaydedilir. Tablo oluşturulurken mutlaka Primary Key alan belirlenmelidir.

Görsel 6.33'te müşteri bilgilerinin kayıtlı olduğu örnek bir tablo verilmiştir.

MusteriID	SirketAdi	Musteri...	MusteriUnvani	Adres	Sehir	PostaKodu	Ulke	Telefon	Faks
ALFKI	Alfreds F...	Maria An...	Sales Represent...	Obere St...	Berlin	12209	Germany	030-00...	030-007...
ANATR	Ana Trujil...	Ana Trujil...	Owner	Avda. d...	México...	05021	Mexico	(5) 555-...	(5) 555-...
ANTON	Antonio ...	Antonio ...	Owner	Matader...	México...	05023	Mexico	(5) 555-...	*NULL*
AROUT	Around t...	Thomas ...	Sales Represent...	120 Han...	London	WA1 1DP	UK	(171) 5...	(171) 55...
BERGS	Berglund...	Christina...	Order Administ...	Berguvs...	Luleå	S-958 22	Sweden	0921-1...	0921-12...
BLAUS	Blauer Se...	Hanna M...	Sales Represent...	Forsterst...	Mannh...	68306	Germany	0621-0...	0621-08...
BLONP	Blondesd...	Frédériq...	Marketing Man...	24, plac...	Strasbo...	67000	France	88.60.1...	88.60.15...
BOLID	Bólido C...	Martín S...	Owner	C/ Araq...	Madrid	28023	Spain	(91) 55...	(91) 555...
BONAP	Bon app'	Laurence...	Owner	12, rue d...	Marseille	13008	France	91.24.4...	91.24.45...
BOTTM	Bottom-...	Elizabeth...	Accounting Ma...	23 Tsaw...	Tsawass...	T2F 8M4	Canada	(604) 5...	(604) 55...
BSBEV	B's Bever...	Victoria ...	Sales Represent...	Fauntler...	London	EC2 5NT	UK	(171) 5...	*NULL*
CACTU	Cactus C...	Patricio S...	Sales Agent	Cerrito 3...	Buenos...	1010	Argentina	(1) 135-...	(1) 135-...
CENTC	Centro c...	Francisc...	Marketing Man...	Sierras d...	México...	05022	Mexico	(5) 555-...	(5) 555-...
CHOPS	Chop-su...	Yang Wa...	Owner	Hauptst...	Bern	3012	Switzerla...	0452-0...	*NULL*
COMMI	Comérci...	Pedro Af...	Sales Associate	Av. dos ...	Sao Pa...	05432-043	Brazil	(11) 55...	*NULL*
CONSH	Consolid...	Elizabeth...	Sales Represent...	Berkeley...	London	WX1 6LT	UK	(171) 5...	(171) 55...
DRACD	Drachen...	Sven Ottl...	Order Administ...	Walserw...	Aachen	52066	Germany	0241-0...	0241-05...
DUMON	Du mond...	Janine La...	Owner	67, rue d...	Nantes	44000	France	40.67.8...	40.67.89...
EASTC	Eastern C...	Ann Dev...	Sales Agent	35 King ...	London	WX3 6FW	UK	(171) 5...	(171) 55...
ERNSH	Ernst Ha...	Roland ...	Sales Manager	Kirchgas...	Graz	8010	Austria	7675-3...	7675-34...
FAMIA	Familia A...	Aria Cruz	Marketing Assis...	Rua Oró...	Sao Pa...	05442-030	Brazil	(11) 55...	*NULL*
FISSA	FISSA Fa...	Diego Roel	Accounting Ma...	C/ Mora...	Madrid	28034	Spain	(91) 55...	(91) 555...
FOLIG	Folies go...	Martine ...	Assistant Sales ...	184, cha...	Lille	59000	France	20.16.1...	20.16.10...

Görsel 6.33: Örnek bir veri tabanı tablosu

Sıra Sizde

Tablodaki Primary Key olarak belirlenmesi gereken alanı bulunuz.

6.3.1. Tablo Oluşturma

Veri tabanlarında verilerin kaydedilmesi için tablolar oluşturulmalıdır. Her bir nesne için ayrı tablo oluşturulur. Nesne özellikleri, tabloda alan olarak belirtilir. Alanlara ait bazı önemli özellikler şunlardır:

Not Null: Kayıt eklenirken o alanın boş geçilemeyeceğini belirtir.

Primary Key: Alanı birincil anahtar olarak belirler.

Auto_increment: Tabloya yeni bir kayıt eklendiğinde benzersiz bir sayının otomatik olarak oluşturulmasına izin verir. Bu sayı, bir önceki değerin 1 fazlası olur. Genelde "id" için kullanılır.

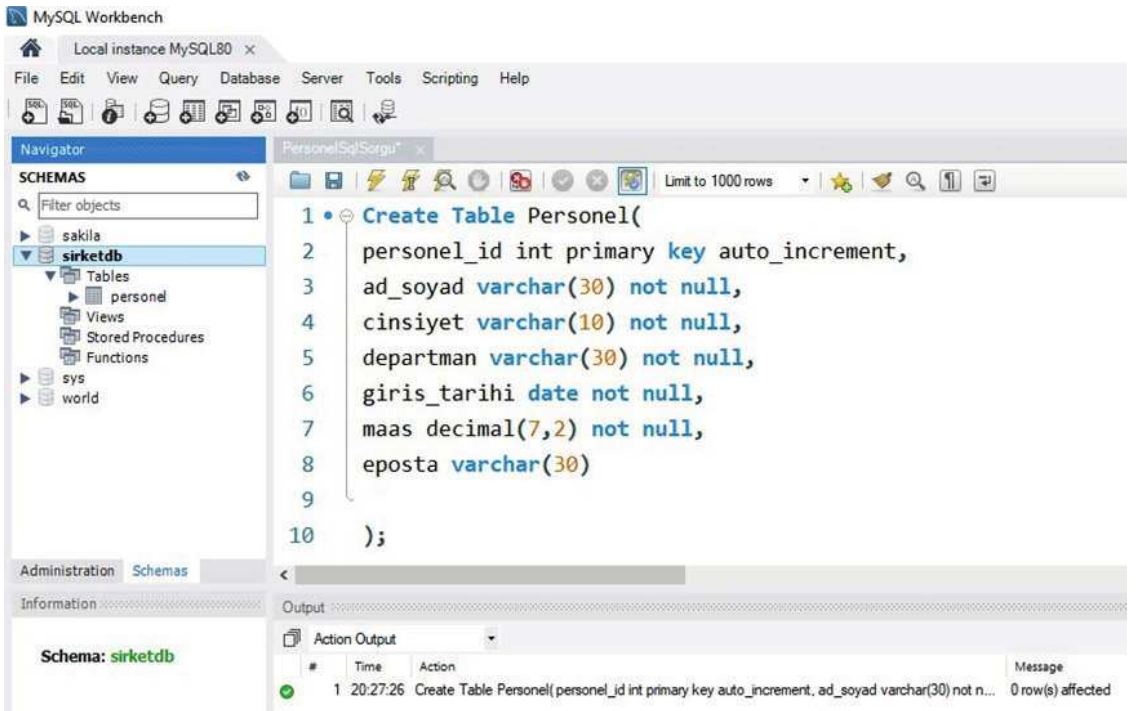
SQL Sorgusuyla Tablo Oluşturma

SQL ile tablo oluşturulurken "Create Table" komutu kullanılır. Eklenecek tablo alanları için veri tipinin yanı sıra o alanda bulunması istenen özellikler de belirtilir. Tablo oluşturma komutu kullanılırken sözdizimine dikkat edilmelidir. Create sorgu yapısı şu şekilde kullanılır:

```
CREATE TABLE tabloAdı (
    alanAdi1 veriTürü özellikler,
    alanAdi2 veriTürü özellikler,
    alanAdi3 veriTürü özellikler,
   ....
);
```

Not Program üzerinde SQL sorgusu çalıştırılacağında hangi veri tabanı üzerinde işlem yapılacaksa o veri tabanının seçili olması gerekir. İşlem yapılmak istenen veri tabanı seçili değilse veri tabanına çift tıklanarak seçili hâle getirilir. Ayrıca sorgu yazmak için simgesi ile yeni sorgu ekranı açılır.

Görsel 6.34'teki SQL sorgusu çalıştırıldığında "Personel" isimli tablo oluşturulur. sirketdb veri tabanı veya onun içinde alt taraftaki Tables bölümü sağ tıklanarak "Refresh All" seçeneği ile yenilenirse oluşturulan tablo o bölümde görüntülenir.



Görsel 6.34: SQL komutlarıyla personel tablosu oluşturma

Tablodaki alanlar incelendiğinde personel_id alanının **int** veri tipinde, PK ve otomatik artan sayı olduğu görülür. Diğer alanlara bakıldığında metinsel ifade olan alanlar, **varchar** veri tipini almış ve maksimum uzunlukları belirtilmiştir. İşe giriş tarihi **date** veri tipinde belirtildiği için bu alana "YYYY-AA-GG" formatında veri girişi yapılmalıdır. Maaş bilgisinin girileceği alan decimal(7,2) olarak belirtilmiştir. Bu alanda 7 rakamı ile toplam uzunluk ve 2 rakamı ile virgülden sonraki basamak sayısı temsil edilir. Bu tabloda yalnızca eposta alanı boş bırakılabilir olarak ayarlanmıştır. Diğer alanlardan herhangi biri boş bırakılırsa veri ekleme işlemi başarısız sayılır.

Program Arayüzüyle Tablo Oluşturma

Arayüzdeki kısaltmaların açılımları şunlardır:

PK: Eklenen satırın Primary Key olup olmamasının seçildiği kutucuktur.

NN: Not Null, eklenen satırın boş geçilemeyeceğini belirtir.

BIN: Seçilen satırdaki bilgilerin Binary şeklinde saklanıp saklanmayacağının seçildiği alandır.

UN: Unsigned, seçilen alan için sadece pozitif ekleme yapılabileceğini gösteren alandır. Örneğin bir tinyint alan unsigned olarak işaretlendiğinde 0 ile 255 arasında bir değer alabilir.

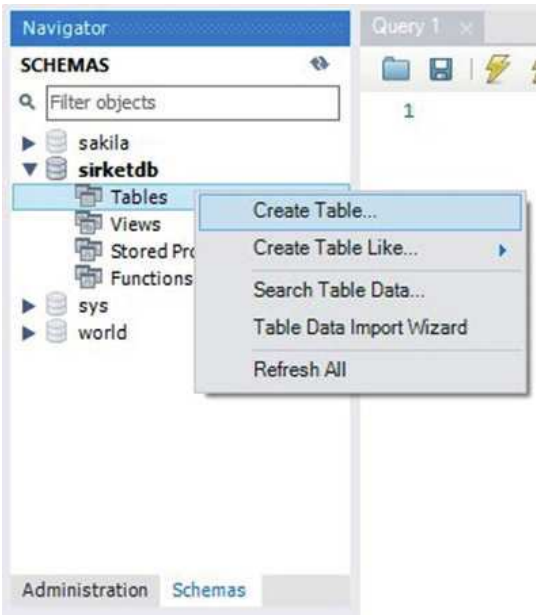
UQ: Seçilen alanın benzersiz olması istendiğinde kullanılır.

ZF: Seçilen alanda boşluk kalması hâlinde alanın "0" ile doldurulması istendiğinde seçilmesi gereken alandır. Örneğin int(3) alanına 21 bilgisi geldiğinde bu alan 021 olarak eklenir.

G: Bu alanın diğer alanlardan üretildiğini gösterir.

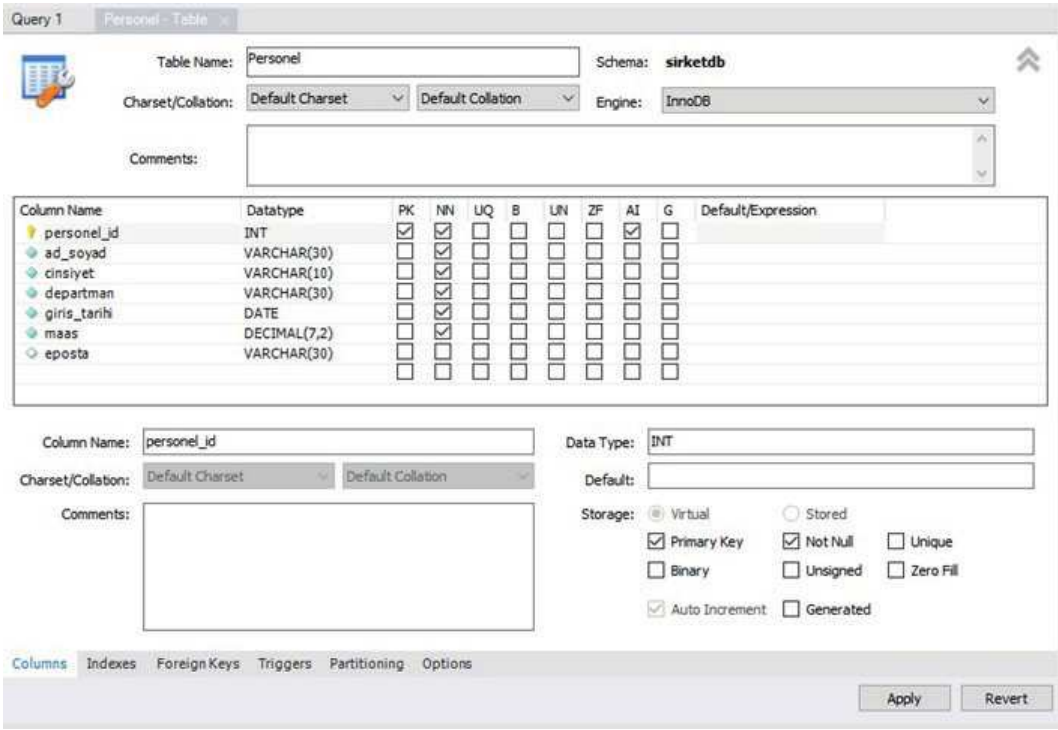
AI: Auto Increment, bu alandaki değerin birer birer artacağını gösterir. Birincil anahtarlar için kullanılır.

Program arayüzüyle tablo oluşturmada öncelikle veri tabanının alt kısmındaki "Tables" bölümüne sağ tıklanarak açılan menüden "Create Table" seçeneği seçilir (Görsel 6.35).



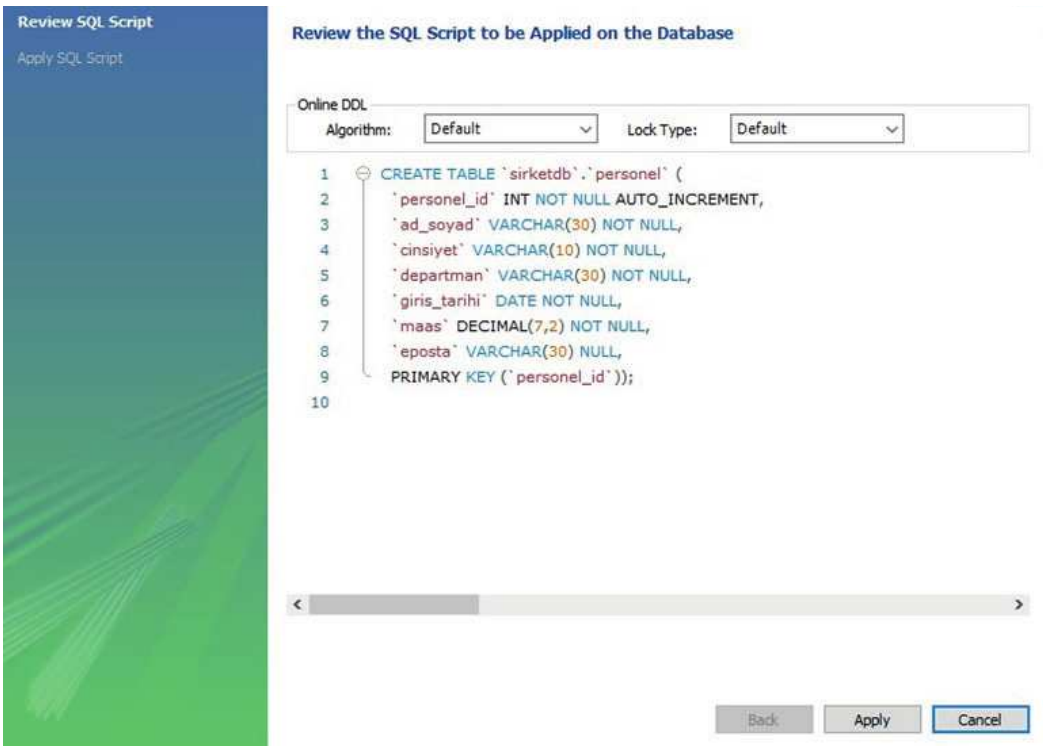
Görsel 6.35: Program arayüzüyle tablo oluşturma

Görsel 6.36'da görülen tablo oluşturma ekranında tablo adı, alan adları ve özellikleri doğru bir şekilde belirlenir. Daha sonra "Apply" butonuna tıklanarak işleme devam edilir.



Görsel 6.36: Personel tablosunun alanlarını ve özelliklerini belirleme

Görsel 6.37'de veri tabanı üzerinde çalışacak SQL sorgusu görülür. Tablo, program arayüzünden oluşturulsa da arka planda bu tasarımın karşılığı olan SQL komutu çalıştırılacaktır. "Apply" butonuna tıklanarak tablonun oluşturulması sağlanır.



Görsel 6.37: Arayüzden oluşturulan tablonun SQL kodları

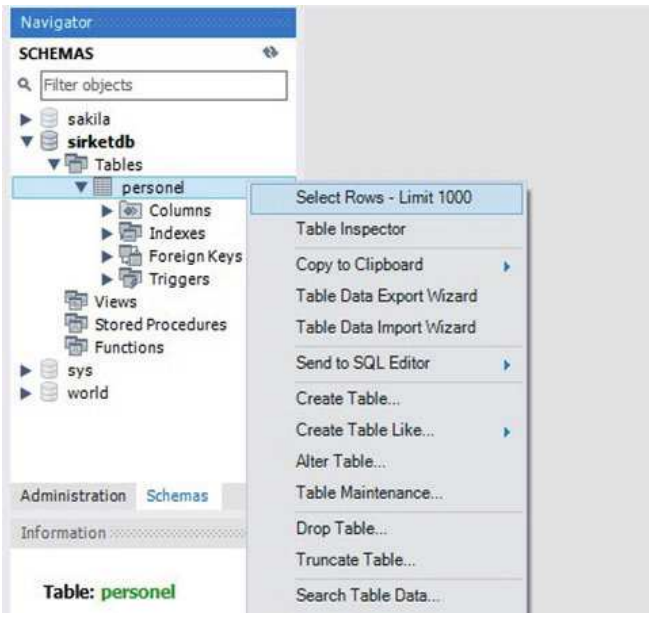
Oluşturulan bir tablonun silinmesi için tablo üzerine sağ tıklanarak **Drop Table** seçeneği seçilir. Ayrıca "Drop Table TabloAdı" komutuyla da silme işlemi yapılabilir.

Oluşturulan tablo üzerinde değişiklik yapılması için tablo üzerine sağ tıklanarak **Alter Table** seçeneği seçilir. Komut ile tablo üzerinde değişiklik yapılmak istenirse "ALTER TABLE "Tablo adı" MODIFY AlanAdı YeniÖzellikler" şeklinde bir komut çalıştırılmalıdır.

6.3.2. Tablolara Veri Girişi

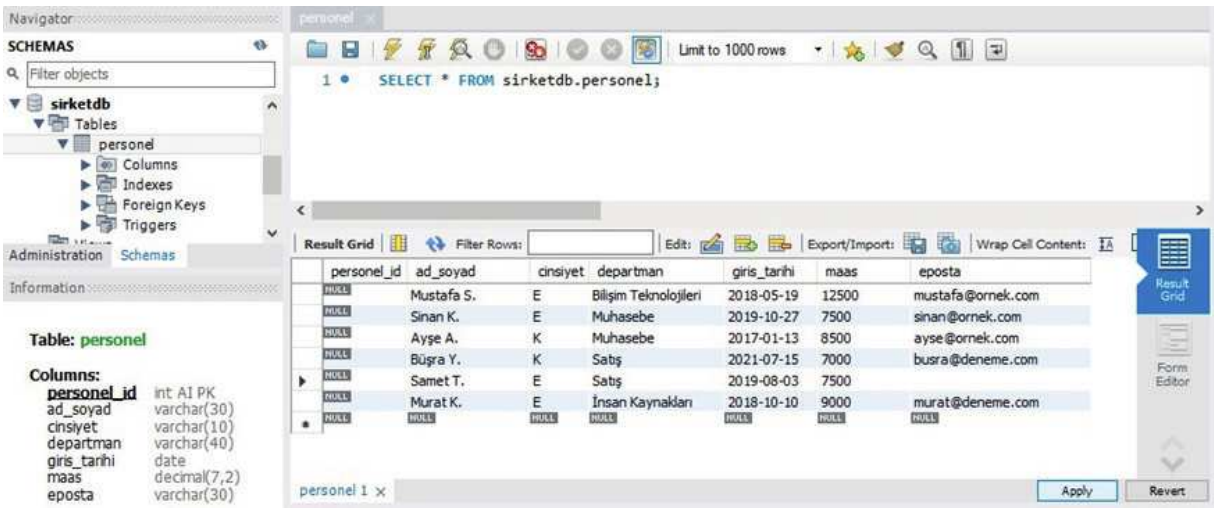
Veri tabanında oluşturulan tablolara veri girişi, program arayüzünden kolaylıkla yapılabilir. Ayrıca SQL komutları kullanılarak da veri girişi yapılır. Bu bölümde program arayüzünden veri ekleme işlemi gerçekleştirilecektir. SQL sorguları konusunda komutlarla veri ekleme işlemi de gösterilecektir.

Veri girişi yapılacak tablonun üzerinde sağ tıklanarak "Select Rows – Limit 1000" seçeneği seçilir. Aslında bu komutla tablodaki 1000 satırlık veri listelenir fakat kullanıcının o tablo içine veri girişi yapmasına da imkân verilir (Görsel 6.38).



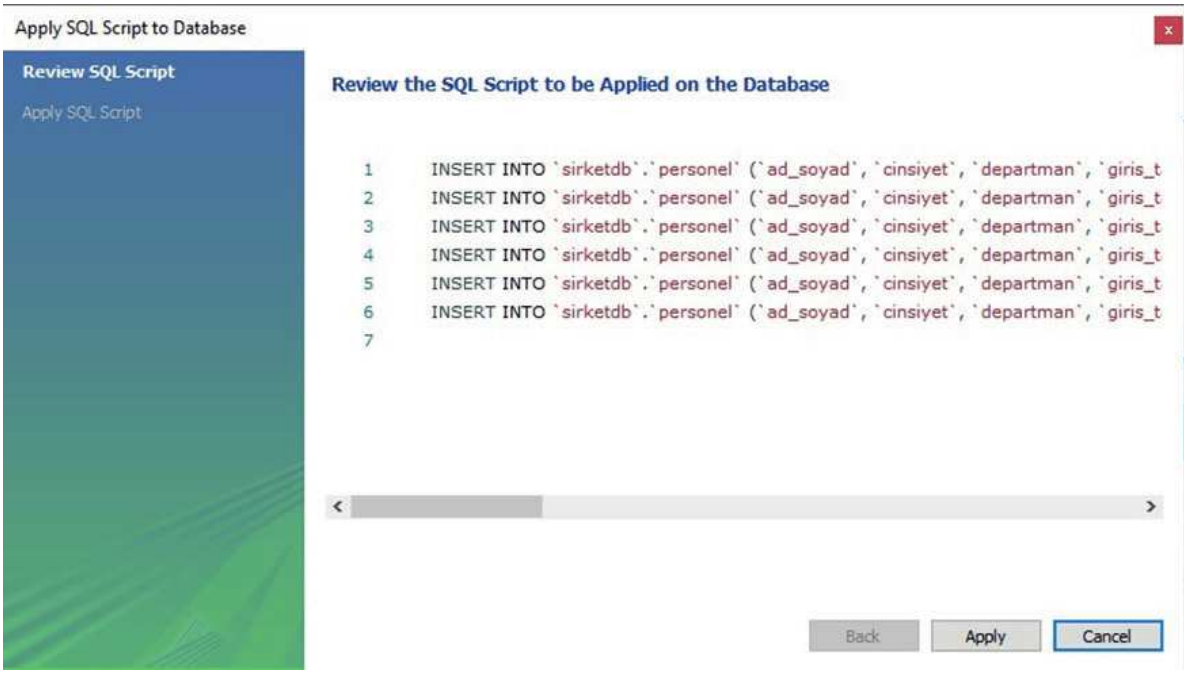
Görsel 6.38: Tablo veri giriş ekranını açma

Alanlara veri girişi yapılırken belirlenen özelliklere dikkat edilmelidir. Görsel 6.38'de görüldüğü gibi **personel_id** alanı otomatik artan olduğu için veri girişi yapılmamıştır. Bu alanın değeri, sistem tarafından otomatik olarak verilir. Diğer alanlara bakıldığında sadece **eposta** alanı boş bırakılabilir olarak ayarlanmıştır. Bu yüzden bu alana değer girilmediğinde hata vermez fakat "NOT NULL" olarak belirlenen diğer alanlara veri girişi yapılmadığında hata oluşur. Veri girişi yapıldıktan sonra "Apply" butonuna tıklanır (Görsel 6.39).



Görsel 6.39: Personel tablosuna veri girişi yapılması

Görsel 6.40'ta eklenecek verilere karşılık gelen SQL komutları verilmiştir. Program, verileri eklemek için bu komutları çalıştırır. "Apply" butonu tıklanarak veri ekleme işlemi tamamlanır.



Görsel 6.40: Personel tablosu veri girişi SQL komutları

Sıra Sizde

sirketdb isimli veri tabanına müşteri bilgilerinin tutulacağı "musteri" isimli bir tablo oluşturunuz. Tabloda müşterinin adı ve soyadı, cinsiyeti, adresi, doğum tarihi, telefonu ve yaşadığı şehir bilgileri tutulacaktır.

6.4. SQL KOMUTLARI

SQL komutları; veri tabanındaki verilerle ilgili ekleme, silme, güncelleme ve listeleme gibi işlemlerin yapılması için kullanılır. Ayrıca veri tabanı, tablo gibi nesneleri oluşturma ve silme işlemleri de SQL komutları ile yapılabilir. Bu bölümde CRUD işlemleri olarak da bilinen ekleme, listeleme (okuma), güncelleme ve silme komutları vardır. Komutların yazımında büyük ve küçük harf duyarlılığı yoktur.

SQL komutları yazılmadan önce simgesine tıklanarak yeni bir SQL sorgu sayfası açılır. Bu sayfada yazılan komutların çalıştırılması için simgesine tıklanır. Aynı sayfada birden fazla komut yazılmışsa ve bunlardan biri çalıştırılmak istenirse simgesine tıklanarak sadece imlecin bulunduğu satırdaki komut çalıştırılabilir.

Sıra Sizde

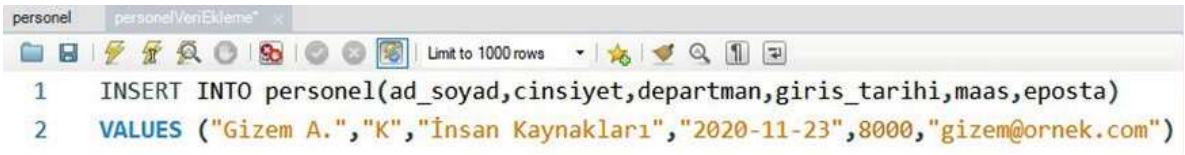
SQL DDL ve SQL DCL komutlarını araştırınız.

6.4.1. INSERT INTO Komutu (Kayıt Ekleme)

Veri tabanı tablolarına veri ekleme işlemi INSERT INTO komutuyla gerçekleştirilir. INSERT INTO komutu şu şekilde kullanılır:

INSERT INTO tabloadı (alan1, alan2, …, alan(n)) **VALUES** (deger1, deger2, …, deger(n))

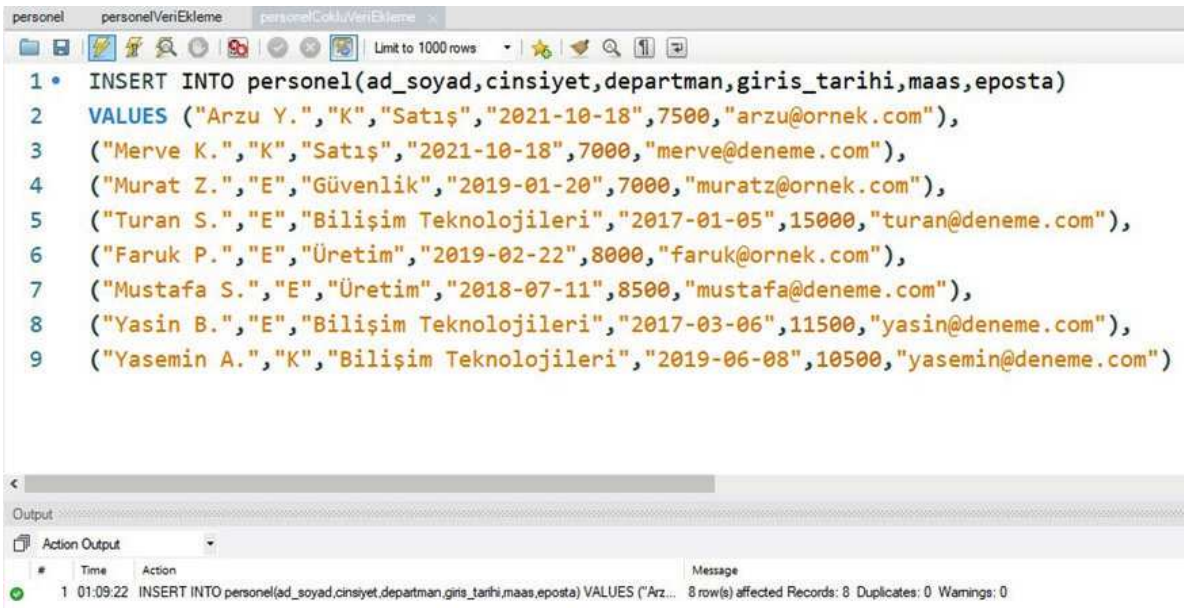
Görsel 6.41'deki komut çalıştırıldığında kaydın personel tablosuna eklendiği görülür. Kontrol etmek için tablo üzerine sağ tıklanarak "Select Rows – Limit 1000" seçeneği seçilir.



Görsel 6.41: Personel tablosuna INSERT INTO komutuyla veri ekleme

Tabloya Tek Sorguda Birden Fazla Veri Ekleme

Bir tabloya bir satırdan fazla verinin aynı anda eklenmesi için "INSERT INTO" komutu Görsel 6.42'deki gibi kullanılır. Sorgu incelendiğinde her satıra eklenecek veriden sonra "," işareti kullanılarak verilerin birbirinden ayrılması sağlanır. Sorgu çalıştırıldığında "Action Output" bölümünde 8 satırın eklendiğini ve sorgunun hatasız çalıştığını belirten bir mesaj gösterilir. Sorgu herhangi bir sebeple yanlış yazılırsa bu bölümde hata mesajıyla karşılaşılır. "Message" bölümü incelenerek hata düzeltilir.



Görsel 6.42: Personel tablosuna "INSERT INTO" komutuyla birden fazla veri ekleme

Sıra Sizde

Oluşturduğunuz "Müşteri" tablosuna "INSERT INTO" komutunu kullanarak tek komutla beş farklı kayıt ekleyiniz.

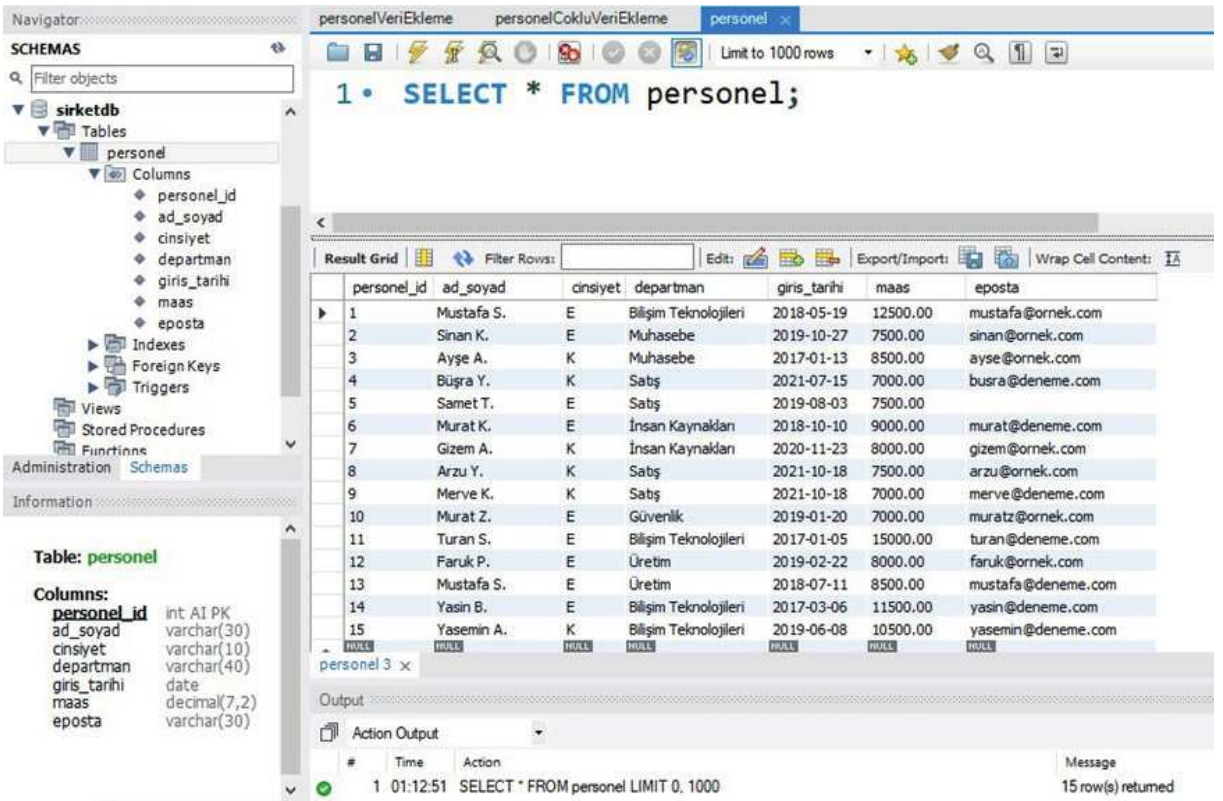
6.4.2. SELECT Komutu (Verileri Listeleme)

Veri tabanı tablolarında verileri listeleme işlemi “SELECT” komutuyla yapılır. SELECT komutu şu şekilde kullanılır:

SELECT Listelenecek Alanlar **FROM** Tablo Adı

Tablodaki bütün alanlar görüntülenmek istenirse Listelenecek alanlar bölümüne “*” yazılır.

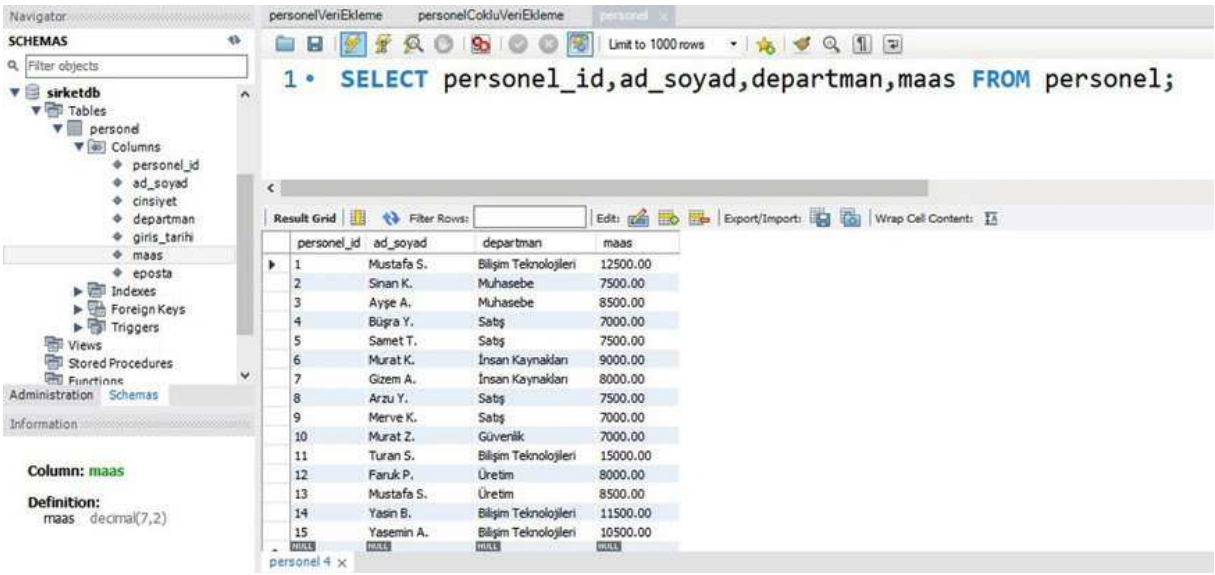
Görsel 6.43’te görüldüğü gibi “SELECT * FROM personel” sorgusu sonucunda tablodaki bütün veriler listelenir. “Action Output” bölümünde sorgunun doğru bir şekilde çalıştığı ve “Message” kısmında ise kaç adet kaydın listelendiği bilgisi görüntülenir.



Görsel 6.43: Personel tablosundaki bütün verileri listeleme

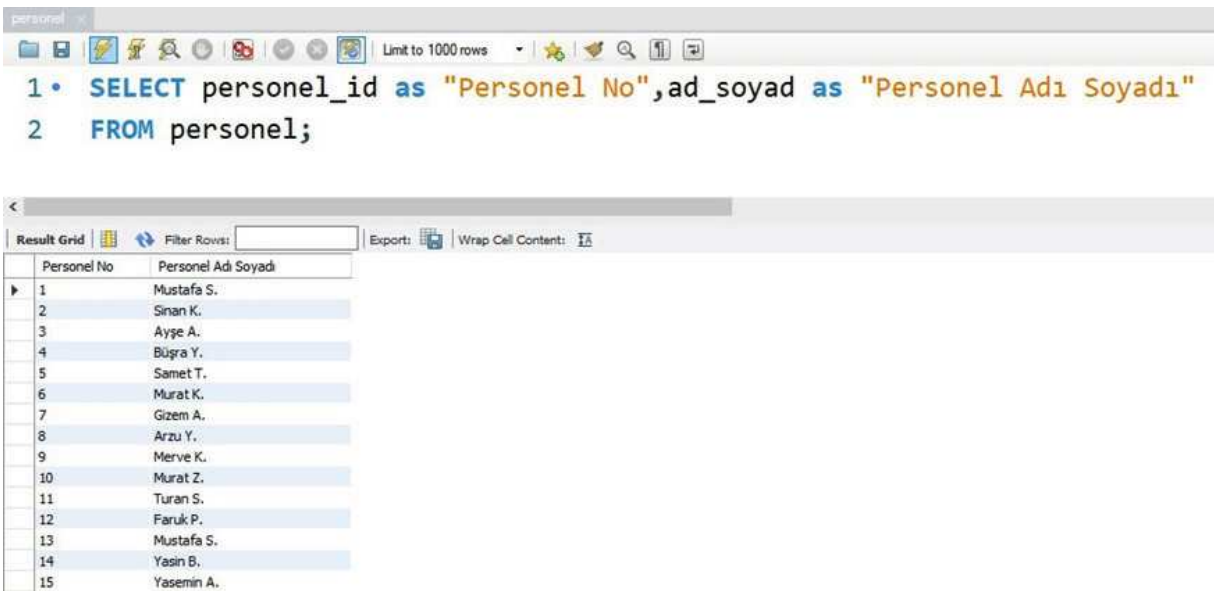
Tabloda Sadece Belirtilen Alanların Listelenmesi

Görsel 6.44'te yazılan sorgu sonucunda bütün kayıtların sadece personel_id, ad_soyad, departman, maaş alanları listelenmiştir. Bunun nedeni, SELECT ifadesinden sonra listelenecek alanlar kısmında bu alanların belirtilmesidir.



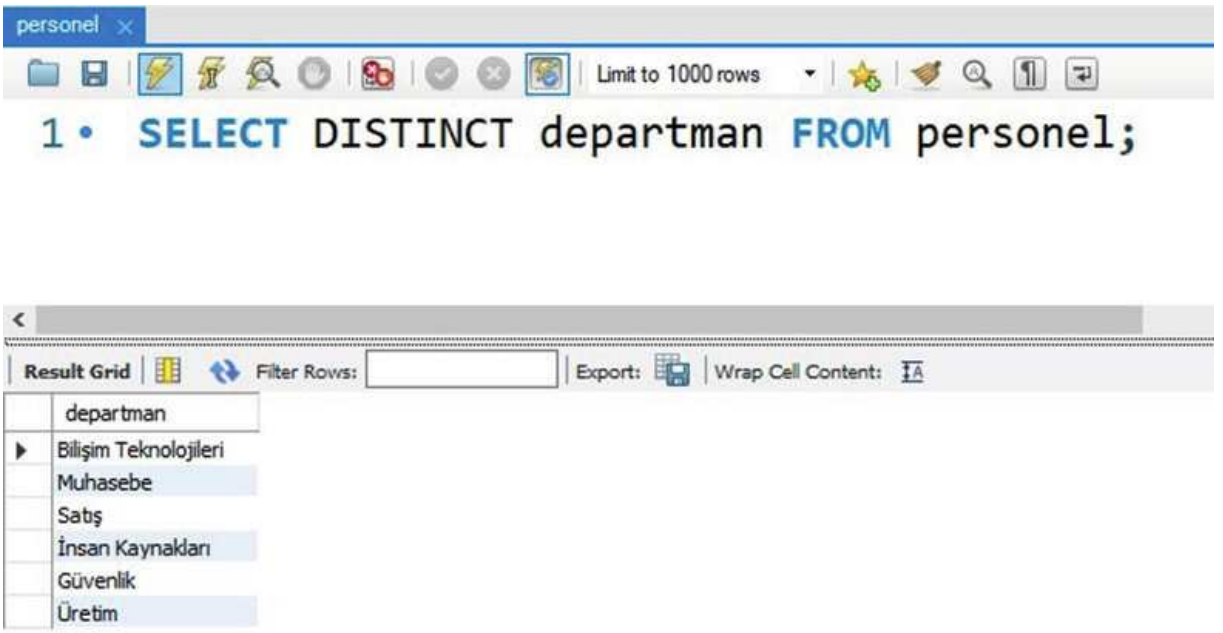
Görsel 6.44: Personel tablosundaki verileri belirtilen alanlarla listeleme

Sütunlarda takma isim kullanılması ve başlıkların alan adları dışında bir isimle görüntülenmesi için "as" anahtar kelimesi yazılır (Görsel 6.45).



Görsel 6.45: Personel tablosunda belirli alanlardaki verileri takma ad kullanarak listeleme

Bir tabloda herhangi bir sütun, yinelenen değerleri içerebilir. Böyle bir durumda tekrarlanan ifadeleri göstermeden farklı değerleri listelemek için **"DISTINCT"** komutu kullanılır (Görsel 6.46).



Görsel 6.46: DISTINCT komutuyla farklı değerleri listeleme

6.4.3. Karşılaştırma Operatörleri

Karşılaştırma operatörleri, iki değer arasında karşılaştırma yapmak için kullanılan operatörlerdir (Tablo 6.2). SQL sorgularında koşul belirtilmek istendiğinde değerler, bu operatörler ile karşılaştırılır (yas>=18, isim="Ali", maas<5000, bolum<>"-Bilişim" vb.).

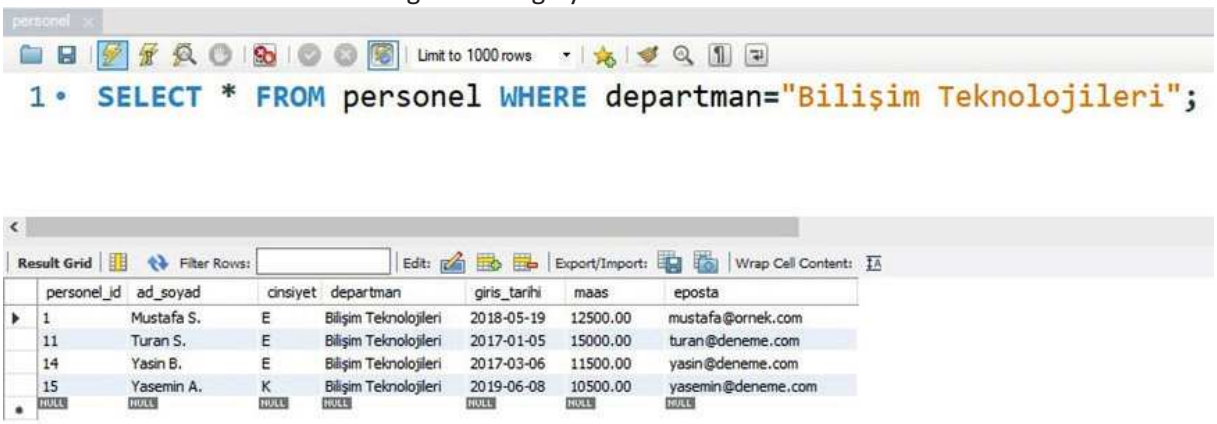
Tablo 6.2: Karşılaştırma Operatörleri

= eşit
> büyük
< küçük
>= büyük eşit
<= küçük eşit
<> farklı (eşit değil)

6.4.4. WHERE Şart İfadesi

SQL sorgularında verilerin filtrelenmesi için "WHERE" komutu kullanılır. Yapılacak işlemin sadece belirli kayıtları etkilemesi istendiğinde bu ifade WHERE komutundan sonra koşul olarak yazılır. Aksi hâlde yapılacak işlemden bütün kayıtlar etkilenir. Listeleme sorgularında herhangi bir problem yaratmayacak bu durum, silme ve güncelleme sorgularında yanlış kayıtların etkilenmesine sebep olur.

Örneğin personel tablosunda yalnızca "Bilişim Teknolojileri" departmanında çalışan personellerin listelenmesi istenirse Görsel 6.47'deki gibi bir sorgu yazılmalıdır.



Görsel 6.47: Bilişim Teknolojileri departmanında çalışan personeli listeleme

Sıra Sizde

1. Cinsiyeti erkek olan personelleri SQL sorgusu kullanarak listeleyiniz.
2. "2018-01-01" tarihinden sonra işe başlayan personellerin ad_soyad, giris_tarihi ve maas alanlarını SQL sorgusu kullanarak listeleyiniz.

BETWEEN Kullanımı

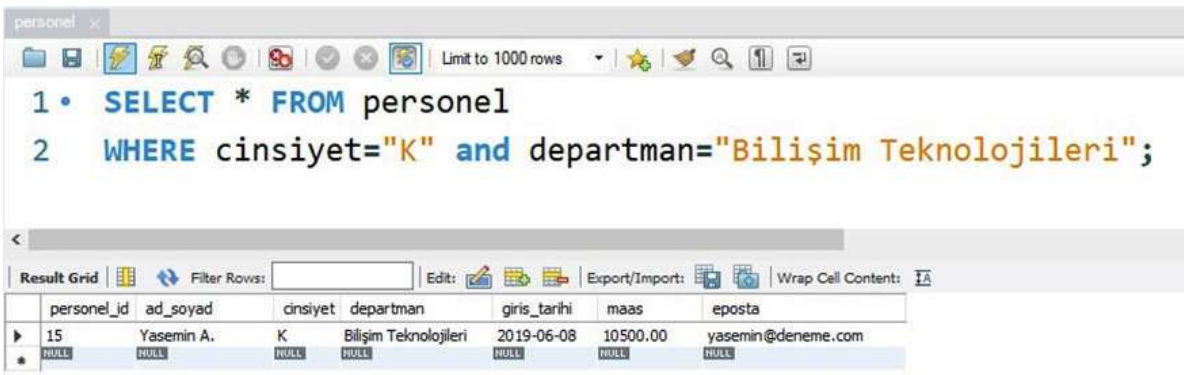
Tablodaki verilerden sadece bir alana göre belirli aralıkta olanlar listelenmek istenirse BETWEEN deyimi kullanılır. Koşul kısmında ilk olarak filtrelenecek alan belirtilir. Ardından "BETWEEN" kullanılarak, aralığın başlangıç ve bitiş değeri "and" operatörü ile ayrılarak yazılır. Örneğin 2021 yılı içinde işe giren personeller listelenmek istenirse sorgu şu şekilde yazılmalıdır:

SELECT * FROM personel WHERE giris_tarihi BETWEEN "2021-01-01" and "2021-12-31"

6.4.5. Mantıksal Operatörler

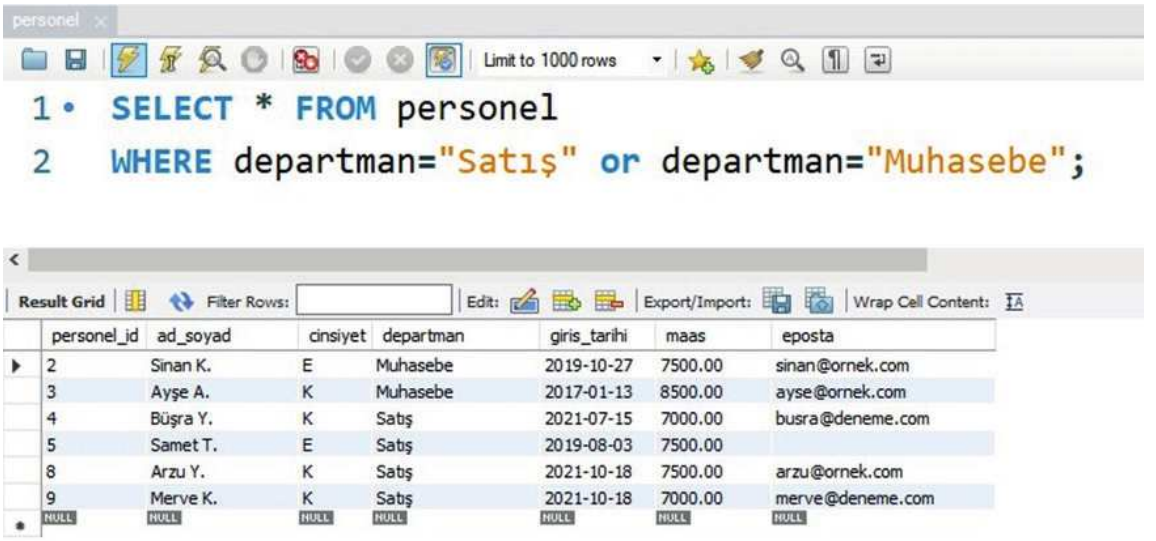
Birden çok koşul ifadesinin olduğu durumlarda koşul ifadelerinin arasında mantıksal operatörler kullanılır.

AND (Ve): Her iki koşulun da sağlanması istenirse **AND** operatörü kullanılır. Örneğin Bilişim Teknolojileri departmanında çalışan kadın personelin listelenmesi istenirse her iki koşulun da sağlanması gerekeceği için koşulların arasında AND operatörü kullanılır (Görsel 6.48).



Görsel 6.48: AND operatörünün kullanımı

OR (Veya): Şartlardan herhangi birinin sağlanması yeterliyse **OR** operatörü kullanılır. Örneğin Satış veya Muhasebe departmanında çalışan personelin listelenmesi istenirse iki koşuldan birinin sağlanması yeterli olacağı için koşulların arasında OR operatörü kullanılır (Görsel 6.49).



personel_id	ad_soyad	cinsiyet	departman	giris_tarihi	maas	eposta
2	Sinan K.	E	Muhasebe	2019-10-27	7500.00	sinan@ornek.com
3	Ayşe A.	K	Muhasebe	2017-01-13	8500.00	ayse@ornek.com
4	Büşra Y.	K	Satış	2021-07-15	7000.00	busra@deneme.com
5	Samet T.	E	Satış	2019-08-03	7500.00	
8	Arzu Y.	K	Satış	2021-10-18	7500.00	arzu@ornek.com
9	Merve K.	K	Satış	2021-10-18	7000.00	merve@deneme.com
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Görsel 6.49: OR operatörünün kullanımı

Sıra Sizde

"2019-11-12" tarihinden önce işe başlayan ve 8500 liradan düşük maaş alan personelleri SQL sorgusu kullanarak listeleyiniz.

6.4.6. Hesaplama Fonksiyonları

Tablolarda listeleme işlemleri yapılırken kayıt sayısını bulma, toplam değeri bulma, ortalama hesaplama, maksimum ve minimum değeri alma gibi bazı işlemlere ihtiyaç duyulabilir. Bu durumda hesaplama fonksiyonları kullanılır.

COUNT Fonksiyonu: Sorgu sonucunda listelenecek kayıt sayısı bu fonksiyon ile belirlenir. Fonksiyonun kullanımı Görsel 6.50'de verilmiştir. Bu fonksiyon, SELECT sorgusunun içinde COUNT (alan adı) şeklinde kullanılır. Kayıt sayısının gösterildiği alana verilecek isim "as" anahtar kelimesiyle belirlenir.

Görsel 6.50: COUNT fonksiyonunun kullanımı

Sıra Sizde

Personel tablosunda "Bilişim Teknolojileri" departmanındaki personel sayısını bulan SQL sorgusunu yazınız.

SUM Fonksiyonu: Seçilen sütundaki kayıtların toplam değeri bu fonksiyon ile hesaplanır. SUM fonksiyonu kullanılırken hangi alandaki değerlerin toplamı hesaplanmak istenirse o alan adı parantez içine yazılır. Görsel 6.51'de yazılan SQL sorgusu çalıştırıldığında personelin toplam maaşı listelenir.

```
SUM
Limit to 1000 rows
1   SELECT SUM(maas) as "Toplam Maaş"
2   FROM personel
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Toplam Maaş
135000.00

Görsel 6.51: SUM fonksiyonunun kullanımı

Sıra Sizde

Personel tablosundaki erkek personelin maaşları toplamını bulan SQL sorgusunuz yazınız.

AVG Fonksiyonu: Seçilen sütundaki kayıtların ortalaması bu fonksiyon ile hesaplanır. AVG fonksiyonu kullanılırken hangi alandaki değerlerin ortalaması hesaplanmak istenirse o alan adı parantez içine yazılır. Görsel 6.52'de görüldüğü gibi kadın personelin ortalama maaşı hesaplanır.

```
AVG*
Limit to 1000 rows
1 •  SELECT AVG(maas) as "Ortalama Maaş"
2    FROM personel WHERE cinsiyet = "K"
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Ortalama Maaş
8083.333333

Görsel 6.52: AVG fonksiyonunun kullanımı

Sıra Sizde

Personel tablosundaki personellerin ortalama maaşını hesaplayan SQL sorgusunu yazınız.

MAX ve MIN Fonksiyonu: Belirtilen alan içindeki en yüksek ve en düşük değer bu fonksiyon ile seçilir. MAX ve MIN fonksiyonları kullanılırken hangi alandaki en yüksek ve en düşük değer gösterilmek istenirse o alan adı parantez içine yazılır. Görsel 6.53'teki sorgu çalıştırıldığında en yüksek personel maaşı gösterilir.

```
MAX*
Limit to 1000 rows
1 •  SELECT MAX(maas) as "En Yüksek Maaş"
2    FROM personel
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

En Yüksek Maaş
15000.00

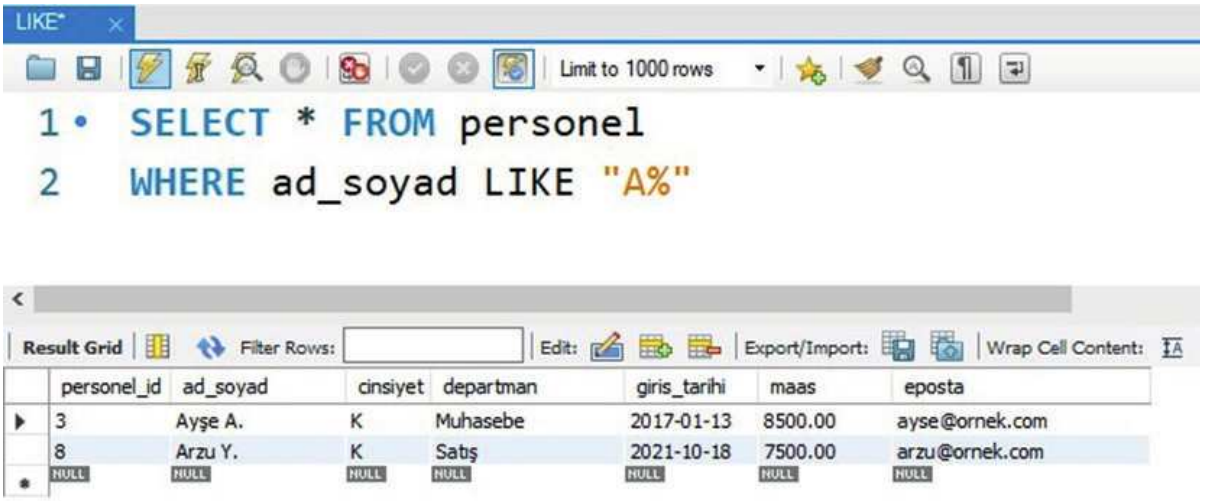
Görsel 6.53: MAX fonksiyonunun kullanımı

Sıra Sizde

Erkek personellerin aldığı maaşlardan en düşük olanını gösteren SQL sorgusunu yazınız.

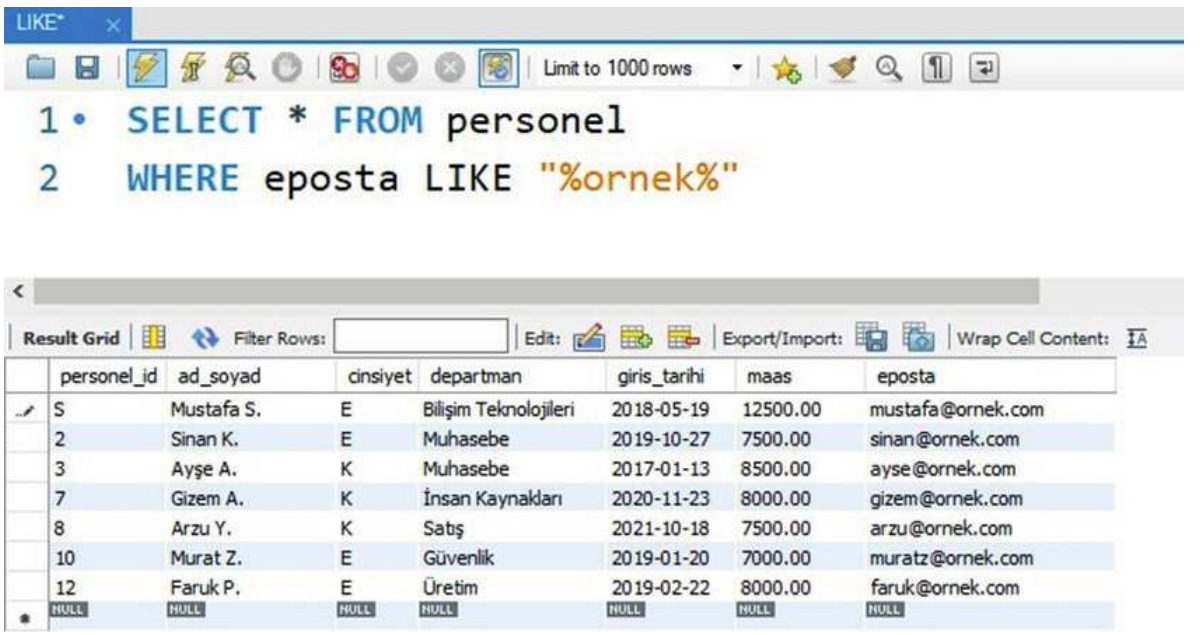
6.4.7. LIKE Komutu (Arama Operatörü)

LIKE operatörü tabloların içindeki veriler üzerinde arama yapmak için kullanılır. Aranacak kelimenin öncesinde veya sonrasında "%" işareti kullanılarak istenen formatta arama yapılabilir. Örneğin A harfi ile başlayan kayıtlar listelenmek istendiğinde sorgudaki koşul kısmı **ad_soyad LIKE "A%"** şeklinde belirtilir. Görsel 6.54'te görüldüğü gibi sorgu sonucunda ad_soyad alanı "A" harfi ile başlayan kayıtlar listelenir.



Görsel 6.54: LIKE operatörünün kullanımı

LIKE komutuyla "ornek" mail hesabına sahip olan kullanıcıların listelenmesine bakıldığında filtreleme işleminin eposta alanına göre yapıldığı görülür. Bu durumda koşul kısmına WHERE eposta LIKE "%ornek%" yazılır. Bu koşul ile "ornek" değerinden önce veya sonra herhangi bir değer gelebileceği anlaşılır (Görsel 6.55).



Görsel 6.55: LIKE operatörüyle arama

Sıra Sizde

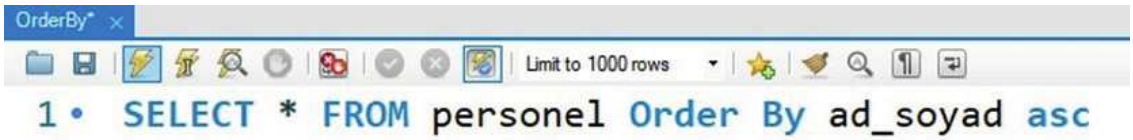
Ad_soyad alanı "M" harfi ile başlayan personelleri listeleyen SQL sorgusunu yazınız.

6.4.8. Order By Komutu (Sıralama)

Tablolardaki verileri sıralamak için SELECT sorgusunda Order By komutu kullanılır. Veriler ASC parametresiyle küçükten büyüğe doğru, DESC parametresiyle de büyükten küçüğe doğru sıralanır. Her iki parametre de kullanılmazsa sıralama işlemi küçükten büyüğe doğru yapılır. SELECT komutunun kullanımı şu şekildedir:

"**SELECT** SeçilecekAlanlar **FROM** TabloAdı **Order By** alanAdı **asc/desc**"

Personelleri isimlerine göre küçükten büyüğe, bir başka deyişle A'dan Z'ye doğru sıralayan SQL sorgusu Görsel 6.56'da verilmiştir. Bu sıralama Z'den A'ya, bir diğer ifadeyle büyükten küçüğe doğru yapılırsa "asc" parametresi yerine "desc" parametresi kullanılır.



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

personel_id	ad_soyad	cinsiyet	departman	giris_tarihi	maas	eposta
8	Arzu Y.	K	Satış	2021-10-18	7500.00	arzu@ornek.com
3	Ayşe A.	K	Muhasebe	2017-01-13	8500.00	ayse@ornek.com
4	Büşra Y.	K	Satış	2021-07-15	7000.00	busra@deneme.com
12	Faruk P.	E	Üretim	2019-02-22	8000.00	faruk@ornek.com
7	Gizem A.	K	İnsan Kaynakları	2020-11-23	8000.00	gizem@ornek.com
9	Merve K.	K	Satış	2021-10-18	7000.00	merve@deneme.com
6	Murat K.	E	İnsan Kaynakları	2018-10-10	9000.00	murat@deneme.com
10	Murat Z.	E	Güvenlik	2019-01-20	7000.00	muratz@ornek.com
1	Mustafa S.	E	Bilişim Teknolojileri	2018-05-19	12500.00	mustafa@ornek.com
13	Mustafa S.	E	Üretim	2018-07-11	8500.00	mustafa@deneme.com
5	Samet T.	E	Satış	2019-08-03	7500.00	
2	Sinan K.	E	Muhasebe	2019-10-27	7500.00	sinan@ornek.com
11	Turan S.	E	Bilişim Teknolojileri	2017-01-05	15000.00	turan@deneme.com
15	Yasemin A.	K	Bilişim Teknolojileri	2019-06-08	10500.00	yasemin@deneme.com
14	Yasin B.	E	Bilişim Teknolojileri	2017-03-06	11500.00	yasin@deneme.com
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Görsel 6.56: Order By komutuyla sıralama

Sıra Sizde

1. Personelleri maaşlarına göre büyükten küçüğe doğru sıralayan SQL sorgusunu yazınız.
2. Personelleri işe başlama tarihine göre (işe önce başlayandan sonra başlayana doğru) sıralayan SQL sorgusunu yazınız.

6.4.9. UPDATE Komutu (Veri Güncelleme)

Tablolardaki verileri güncellemek için UPDATE komutu kullanılır. UPDATE komutu şu şekilde kullanılır:

UPDATE TabloAdı SET alanAdı=yeniDeğer WHERE Koşul İfadeleri

Görsel 6.57'de personel tablosundaki "Arzu Y." isimli personele ait olan "8" numaralı id'ye sahip personelin maaş bilgisini 9500 lira olarak güncelleyen SQL sorgusu verilmiştir. Yalnızca bir personelin maaş bilgisi değiştirileceği için güncelleme işlemi Primary Key alanına göre yapılır. Bu yüzden koşul kısmına "Arzu Y." isimli personelin personel_id değeri yazılır.

Görsel 6.57: UPDATE komutunun kullanımı

Not Yazılan sorguda koşul ifadesinin yazılmadığı düşünülürse "UPDATE personel SET maas=9500" şeklinde bir sorguyla karşılaşılır. Bu durumda koşul belirtilmediği için tablodaki personellerin tamamının maaşı 9.500 lira olarak güncellenir. Binlerce personel olduğu düşünülürse bu durumda çok büyük bir problemle karşılaşılabilir. Binlerce personelin maaş bilgisinin tekrar sisteme girilmesi gerekebilir.

Uyarı : MySQL Workbench programında "WHERE" ifadesi Primary Key alanı ile sınırlandırılmazsa güvenlik sebebiyle güncelleme işlemine izin verilmez çünkü burada yapılacak bir hatayla yüz binlerce kayıt bulunan veri tabanının tamamı etkilenebilir. Bu bir güvenlik önlemidir. Bu güvenlik önlemi "Edit - Preferences - SQL Editor - Safe Updates" seçeneği kapatılarak kaldırılabilir.

Not Silme ve güncelleme ile ilgili güvenlik önlemi devre dışı bırakıldıktan sonra sorgular daha dikkatli yazılmalıdır.

Görsel 6.58'de personel tablosundaki tüm personelin maaşına %20 oranında zam yapılmasını sağlayan SQL ifadesi verilmiştir.

Görsel 6.58: UPDATE komutuyla personel maaşına zam yapma

Sorgu çalıştırıldıktan sonra veriler tekrar listelendiğinde personel maaşlarına %20 oranında zam yapıldığı görülür.

Not Güncelleme ve silme sorgularında bir kayıt için işlem yapılacaksa PK alanına göre koşul belirtilir. Koşul, birden fazla kayıt için silme veya güncelleme işlemi yapılacaksa istenen alanın değerine göre belirtilir.

Sıra Sizde

1. "Bilişim Teknolojileri" departmanının ismini "Bilgi Teknolojileri" şeklinde güncelleyen SQL sorgusunu yazınız.
2. Maaşı 10.000 liranın altında olan personelin maaşına %5'lik ek zam yapılmasını sağlayan SQL sorgusunu yazınız.
3. Cinsiyet alanındaki "E" değerini "Erkek" olarak güncelleyen SQL sorgusunu yazınız.
4. Cinsiyet alanındaki "K" değerini "Kadın" olarak güncelleyen SQL sorgusunu yazınız.

6.4.10. DELETE Komutu (Veri Silme)

Veri tabanı tablolarından verileri silmek için DELETE komutu kullanılır. DELETE komutu şu şekilde kullanılır:

DELETE FROM TabloAdı **WHERE** Koşul ifadeleri

Not DELETE komutu kullanılırken koşul belirtilmezse tablodaki bütün veriler silinir. Örneğin "DELETE FROM Personel" komutu koşulsuz bir silme işlemi yapacağı için personel tablosundaki bütün kayıtlar silinir. Bu yüzden DELETE komutu kullanılırken mutlaka koşul belirtilir.

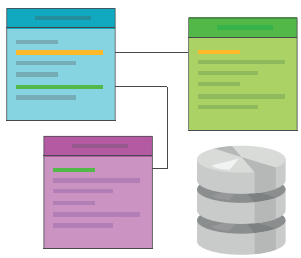
Görsel 6.59'da "Murat K." isimli İnsan Kaynakları personelini silen SQL ifadesi gösterilmiştir. Silme işlemi kaydın PK alanı olan personel_id değerine göre yapılmıştır.

Görsel 6.59: DELETE komutunun kullanımı

Sıra Sizde

Üretim departmanında çalışan bütün personelin silinmesini sağlayan SQL sorgusunu yazınız.

6.5. İLİŞKİSEL VERİ TABANI (RELATIONAL DATABASE)



Görsel 6.60: İlişkisel veri tabanı

İlişkisel veri tabanı, birbirileriyle ilişkili verilerin bulunduğu tablolar arasında bağlantı kurularak tasarlanan bir veri tabanı türüdür (Görsel 6.60). Tablolar birbirlerine **anahtar (PK ve FK)** adı verilen nesnelerle bağlanır. Bu tabloların verileri, birbirlerini referans alır. Bu ilişkisel yapı ile aynı anda birçok tablodan veri çekilmesine olanak sağlanır.

İlişkisel veri tabanı modelinin amacı, yüksek verimliliktir. Bu model ile verilerin bir arada bulunması ve tutarlı olması sağlanır. Ayrıca Normalizasyon kuralları uygulanan bir ilişkisel veri tabanında veri tekrarından kaçınarak performans artırılır. Özellikle büyük sistemlerde ilişkisel veri tabanının kullanılması, verilerin yönetilmesini kolaylaştırır.

6.5.1. İlişkisel Veri Tabanı Tasarımı

İlişkisel veri tabanında her nesne için ayrı bir tablo oluşturulur ve her tablo sadece bir nesneye ait özellikleri barındırır. İlişkisel veri tabanı tasarımı yapılırken oluşturulacak tablolar ve tablolar arasındaki ilişkiler doğru tespit edilmelidir. İlişkisel veri tabanı tasarımı yapılırken önce veri tabanı ve tablolar oluşturulur.

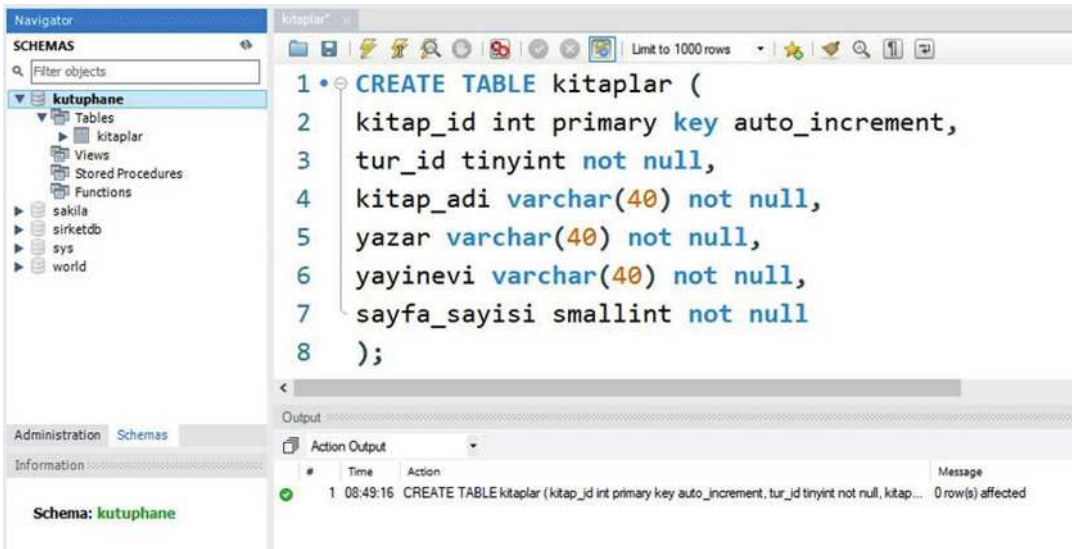
Bu bölümde "Kütüphane Otomasyonu" projesine ait veri tabanının tasarımı yapılacaktır. Veri tabanı oluşturulmadan önce programda yapılacak işlemler ve kaydedilecek bilgiler belirlenir. Projenin amacı, okuldaki öğrencilere kütüphanede kayıtlı kitaplardan ödünç verme işleminin yapılmasıdır. Bunun için

öğrencilerin, kütüphanedeki kitapların ve ödünç verilen kitapların bilgilerini saklayacak tabloların hazırlanması gerekir.

Öncelikle "kutuphane" isminde bir veri tabanı oluşturulur. Veri tabanını oluşturmak için program içinde "**CREATE DATABASE** kutuphane" komutu çalıştırılır (Görsel 6.61).

Görsel 6.61: Kütüphane veri tabanının oluşturulması

Not Veri tabanı üzerinde sorgu çalıştırılacağında işlem yapılacak veri tabanının seçildiğinden emin olunmalıdır. Aksi hâlde sorgu, doğru yazılmış olsa da hata verir. Sol bölümdeki veri tabanına çift tıklanarak veri tabanı seçili hâle getirilir. Sonraki aşamada veri tabanı tabloları oluşturulur. İlk olarak "kitaplar" tablosu oluşturulur (Görsel 6.62). Bu tabloda PK şeklinde belirlenen kitap_id alanı otomatik artan sayı olarak ayarlanmıştır.

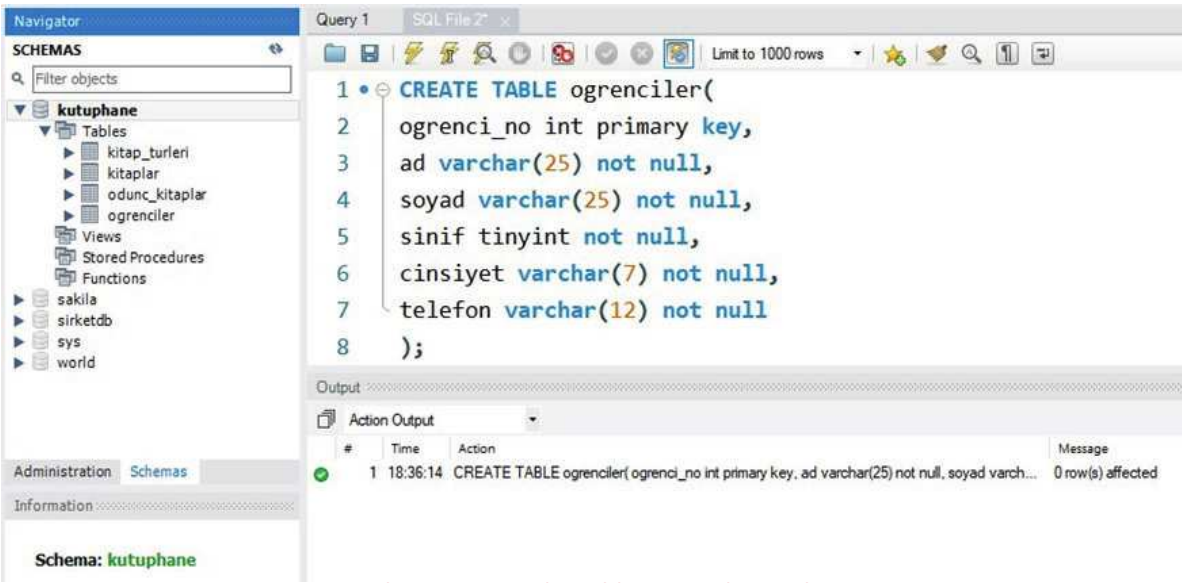


Görsel 6.62: Kitaplar tablosunun oluşturulması

Kitaplar tablosunda kullanılacak tür değerlerini içeren kitap_turleri tablosu oluşturulur (Görsel 6.63). Kitaplar tablosundaki tur_id alanına yalnızca bu tablodaki türlerden biri girilebilir. Böylelikle veri doğruluğu sağlanır.

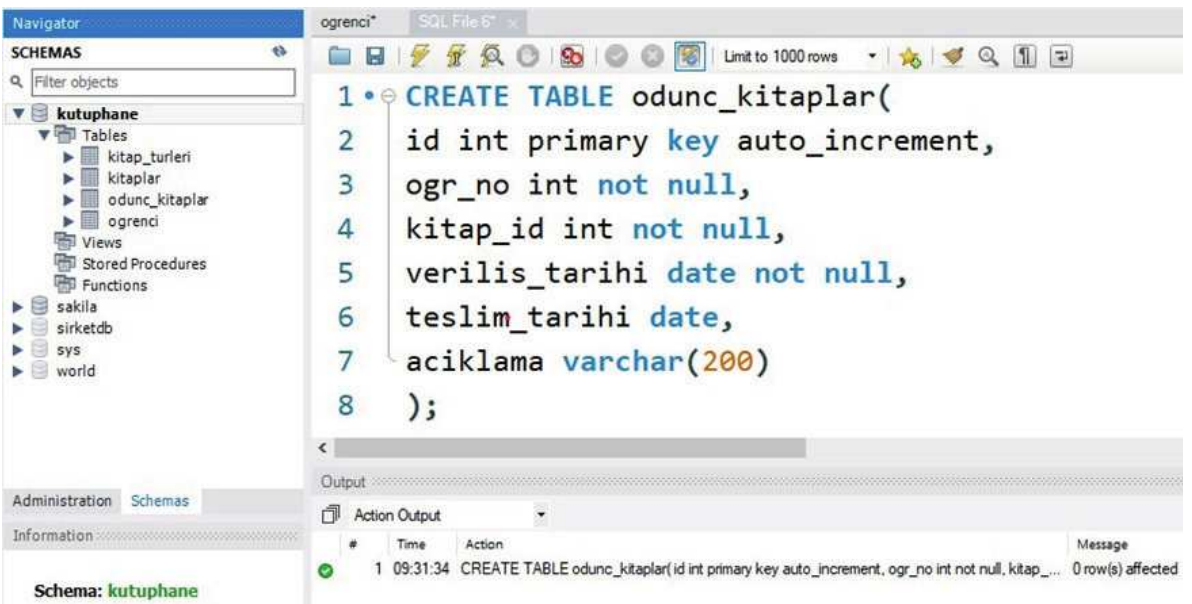
Görsel 6.63: Kitap_turleri tablosunun oluşturulması

Kütüphane projesinde kitapları ödünç alacak öğrenciler de ayrı bir tabloda kayıtlı olacaktır. Bu nedenle öğrenci isimli tablo oluşturulur (Görsel 6.64).



Görsel 6.64: Ogrenciler tablosunun oluşturulması

Son olarak projede öğrencilerin ödünç kitap alma bilgilerini tutan bir tablo oluşturulur (Görsel 6.65). Bu tabloyla sistemde kayıtlı öğrencilerin, sistemde kayıtlı olan kitaplardan birini alması sağlanır. Bu yüzden öğrencilerin ödünç kitap alma bilgilerini tutan tablo, öğrenci ve kitap bilgilerini bu tablolardan referans almalıdır.



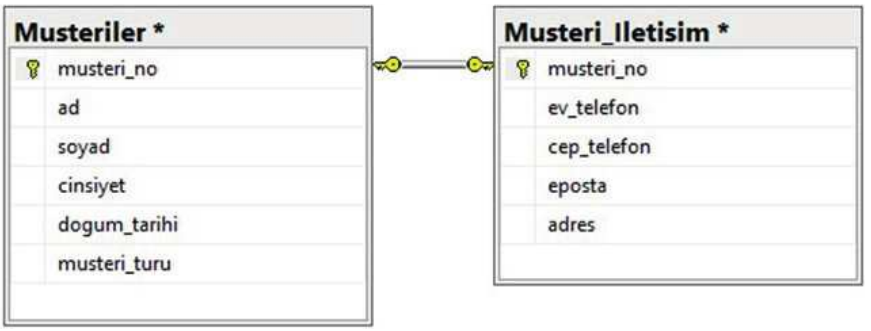
Görsel 6.65: Odunc_kitaplar tablosunun oluşturulması

6.5.2. Tablolar Arası İlişkiler

Veri tabanında tablolar arasında bire bir (one to one), bire çok (one to many), çoka çok (many to many) olmak üzere üç çeşit ilişki türü vardır.

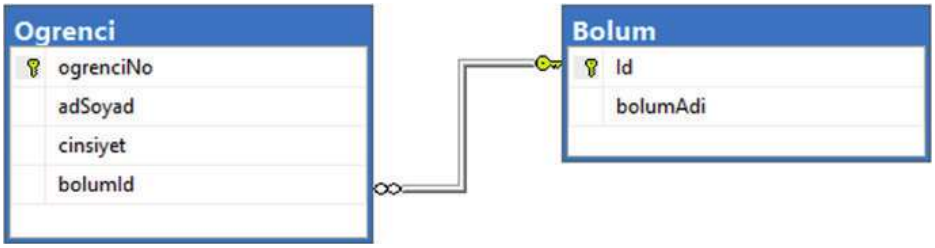
Bire Bir Bağlantı Türü (1:1): Bağlantı kurulacak alandaki kayıt her iki tabloda da yalnızca bir tane mevcut ise bu tablolar arasında bire bir ilişki kurulur. Bu ilişki türünde kullanılacak alanın iki tabloda da PK olarak ayarlanabilmesi gerekir.

Örnek olarak Görsel 6.66'daki müşteri ile müşterinin iletişim bilgilerinin birbirinden ayrıldığı tablolar incelenebilir.



Görsel 6.66: One to one ilişki örneği

Bire Çok Bağlantı Türü (1:n): Bir tabloda Primary Key olan alanın başka bir tabloda birden fazla tekrar etmesiyle ortaya çıkan bağlantı türüdür. Görsel 6.67'deki tablolar incelendiğinde Bölüm tablosunda bölüm değeri yalnızca bir kez bulunabilirken öğrenci tablosunda birçok kez tekrar edebilir çünkü o bölüme ait birden fazla öğrenci bulunabilir. Oluşturulan bire çok ilişki sonrasında "Ogrenci" tablosundaki "bolumId" alanı Foreign Key olarak ayarlanmıştır. Bu alana veri girişi yapılırken yalnızca "Bolum" tablosunda bulunan "Id" değerlerinden biri girilebilir. Aksi hâlde veri girişi başarısız olur.

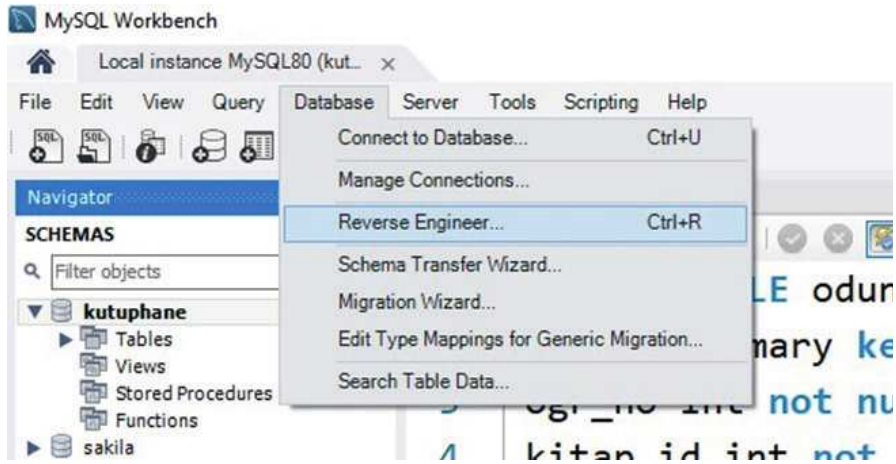


Görsel 6.67: One to many ilişki örneği

Çoka Çok Bağlantı Türü (n:m): İki tablodaki verilerin karşılıklı olarak diğer tablodaki birden fazla veriye karşılık gelmesidir. Bu ilişki en karmaşık ilişki türüdür. Çoka çok ilişkiyi açıklamak için üçüncü bir tabloya ihtiyaç vardır. Arasında çoka çok ilişki olan iki tablo, üçüncü bir tablo ile bire çok ilişki yapılarak kullanılır. Görsel 6.68'deki örnek incelendiğinde bir aktör birden fazla filmde rol alabilir. Aynı şekilde bir filmde de birden fazla aktör oynayabilir. Bu yüzden bu iki tablo arasında çoka çok bir ilişki olmalıdır.

Görsel 6.68: Many to many ilişki örneği

MySQL Workbench programında tablolar arasında ilişki kurmak için "Database" menüsünden "Reverse Engineer" seçeneği seçilir (Görsel 6.69).



Görsel 6.69: Reverse Engineer menüsü

Reverse Engineer menüsü seçildikten sonra Görsel 6.70'teki ekranla karşılaşılır. "Stored Connection" bölümünde kayıtlı bağlantı seçilerek "Next" düğmesine tıklanır.

Görsel 6.70: Veri tabanı bağlantı bilgileri

Görsel 6.71'deki ekranda işlem yapılmak istenen veri tabanı seçilerek "Next" düğmesi ile işleme devam edilir.

Görsel 6.71: Veri tabanı seçimi

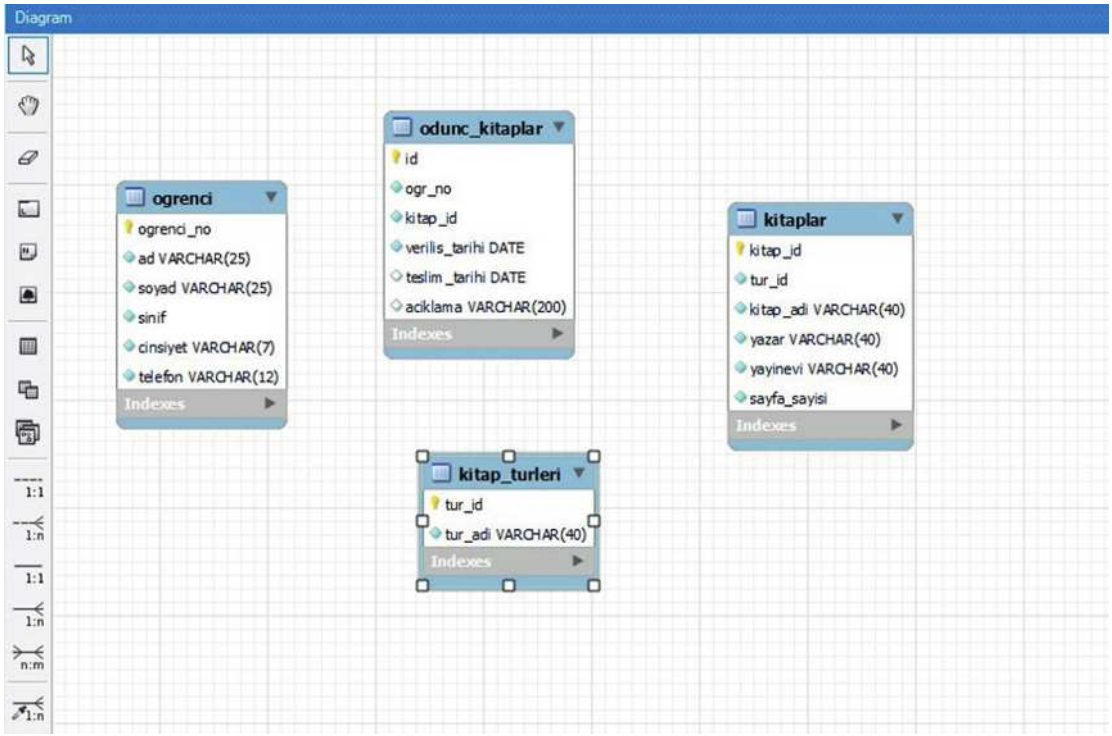
Sonraki aşamada Görsel 6.72'deki ekrandan veri tabanı tabloları seçilir ve "Execute" düğmesine tıklanır.

Görsel 6.72: Veri tabanı tabloları seçimi

Görsel 6.73'te işlemin başarılı olduğunu gösteren bir mesaj görülür. "Next" ve ardından "Finish" düğmesine tıklanarak işlem bitirilir.

Görsel 6.73: İşlemin başarılı olduğunu belirten mesaj ekranı

Görsel 6.74'te görüldüğü gibi eklenen tablolar Diagram ekranında görüntülenir.



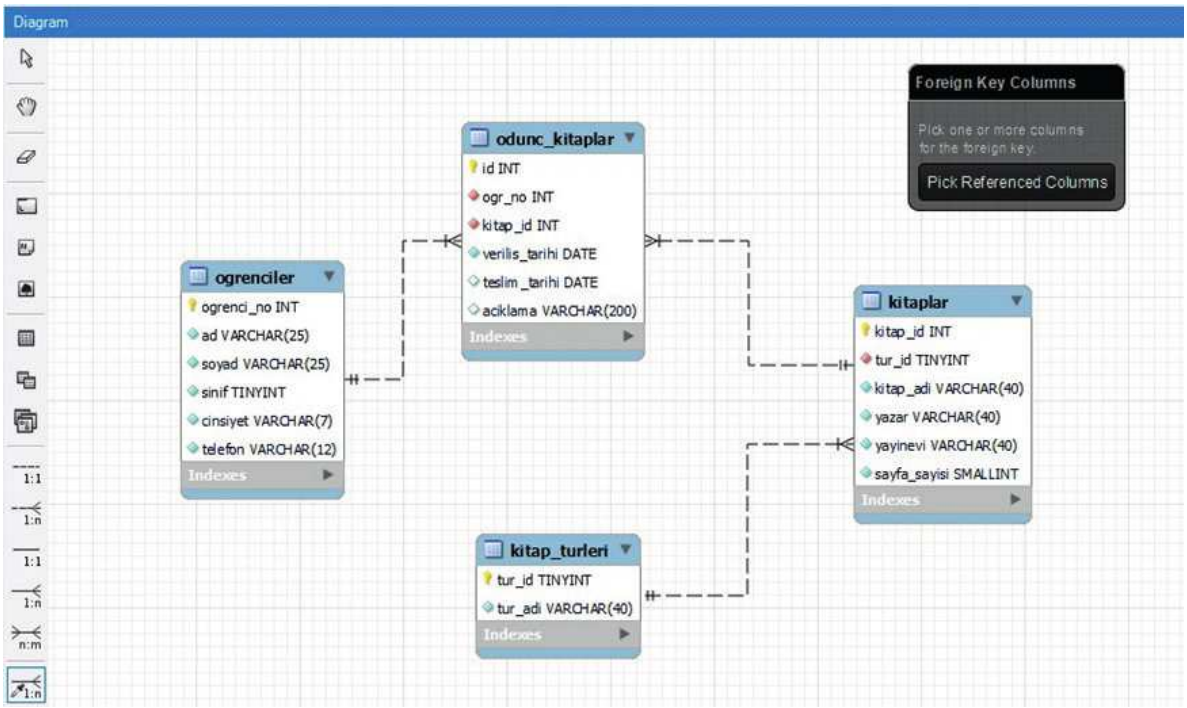
Görsel 6.74: İlişkilerin yapılacağı Diagram ekranı

Dikkat : Bazı durumlarda ekrana veri türleri (INT gibi) gelmez. Veri türleri ekrana gelmemişse tablo başlıklarına çift tıklandıktan sonra açılan pencereden tekrar veri türü tanımlanmalıdır.

Bu veri tabanında bire çok ilişki türü kullanılacağı için Görsel 6.74'teki ekranın sol alt tarafında bulunan simgesine tıklanır. Birden fazla bire çok ilişkiyi temsil eden seçenek varken sol altta bulunan simgenin seçilme amacı, var olan bir Foreign Key alanı üzerinden bağlantı yapılacak olmasıdır.

simgesi tıklandıktan sonra Görsel 6.75'teki ekranla karşılaşılır. Bu ekranda ilişki oluşturmak için ilk olarak Foreign Key alanın daha sonra da Primary Key alanın seçilmesi gerekir.

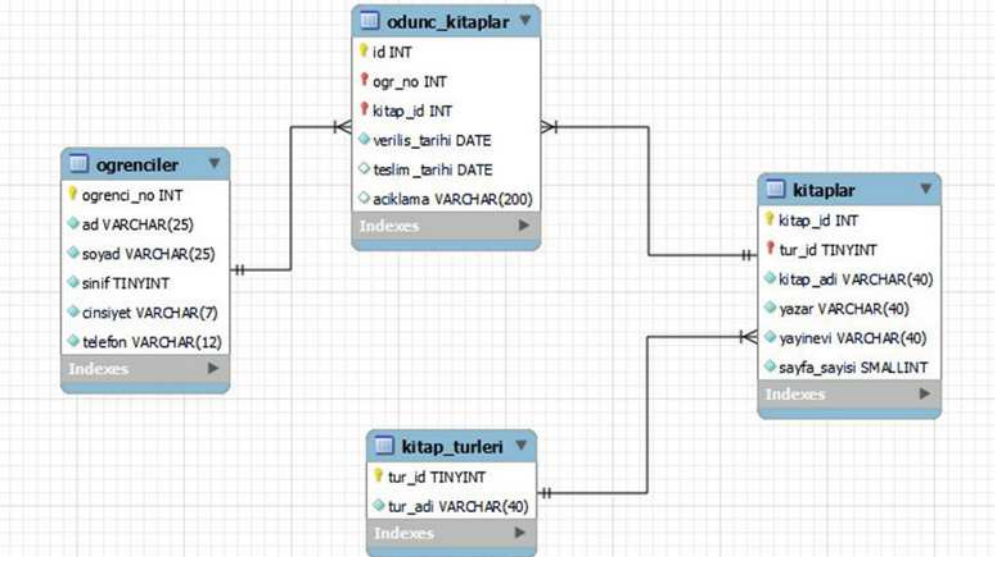
"Kitaplar" tablosundaki FK alanı olan **tur_id** alanı seçildikten sonra "kitap_turler"i tablosundaki PK alanı olan **tur_id** alanı seçilir. Daha sonra simgesine tekrar tıklanarak önce "odunc_kitaplar" tablosunda FK alanı olan **ogr_no** sonra "ogrenciler" tablosundaki PK alanı olan **öğrenci_no** seçilir. Son olarak da simgesine tıklanarak önce "odunc_kitaplar" tablosunda FK alanı olan **kitap_id** sonra "kitaplar" tablosunda PK alanı olan **kitap_id** seçilir ve işlem tamamlanır.



Görsel 6.75: Tablolar arası ilişki oluşturma

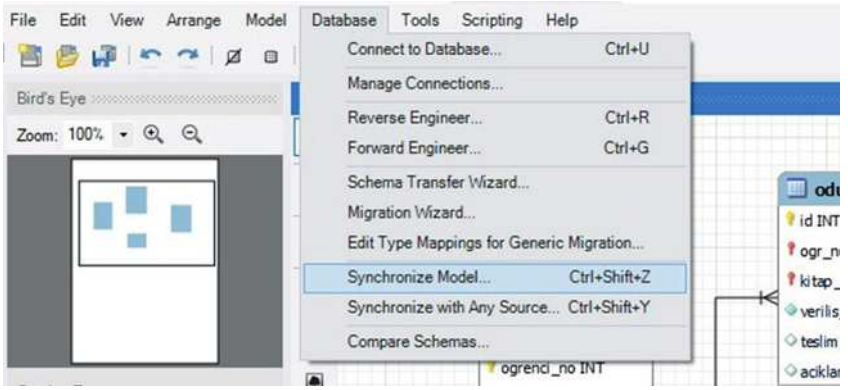
Oluşturulan ilişkinin üzerine gelinerek tablolardaki hangi alanların arasında ilişki kurulduğu kontrol edilir. Bağlantılar tanımlanmadığı için kesik çizgi ile oluşturulmuştur. Bu bağlantı türü uygun değildir. İlişkiyi tanımlamak için oluşturulan bağlantıya çift tıklandıktan sonra alt tarafta "Relationship" penceresinde "Foreign Key" sekmesine geçilir. Daha sonra **Identifying Relationship** kutucuğu seçilir.

Bu işlem yapıldıktan sonra aradaki bağlantının kesik çizgi değil, düz çizgi şeklinde oluştuğu görülür (Görsel 6.76). Bu işlem, bütün ilişkiler için yapılır. İlişkiyi tanımlama işlemi sonucunda tablolara veri girişi yapılırken FK olan alan, bağlantılı olduğu tablodaki değerler dışında veri girişi kabul etmez.



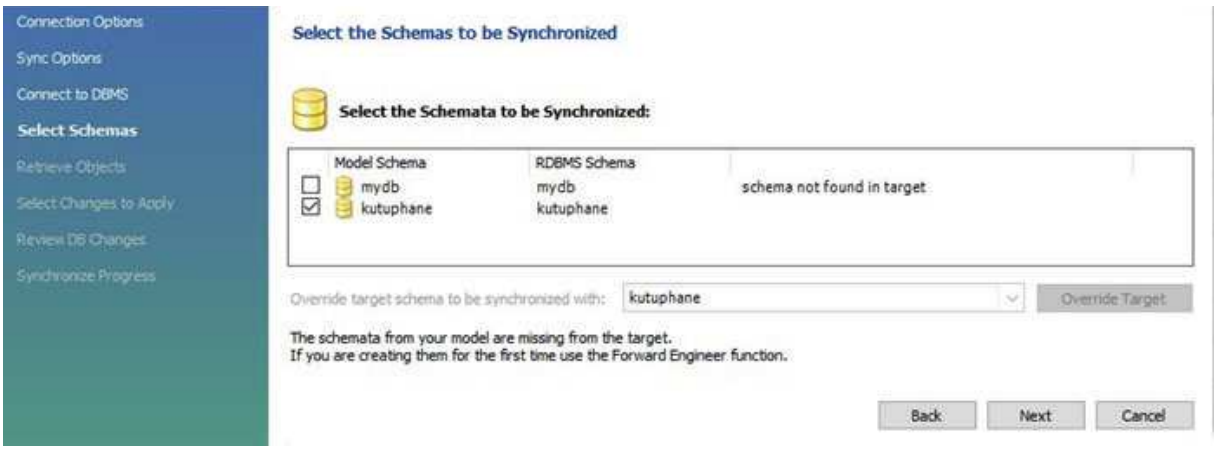
Görsel 6.76: İlişkileri tanımlama

Yapılan değişikliklerin veri tabanına uygulanması için "Database" menüsünden "Synchronize Model" seçeneği işaretlenir (Görsel 6.77).



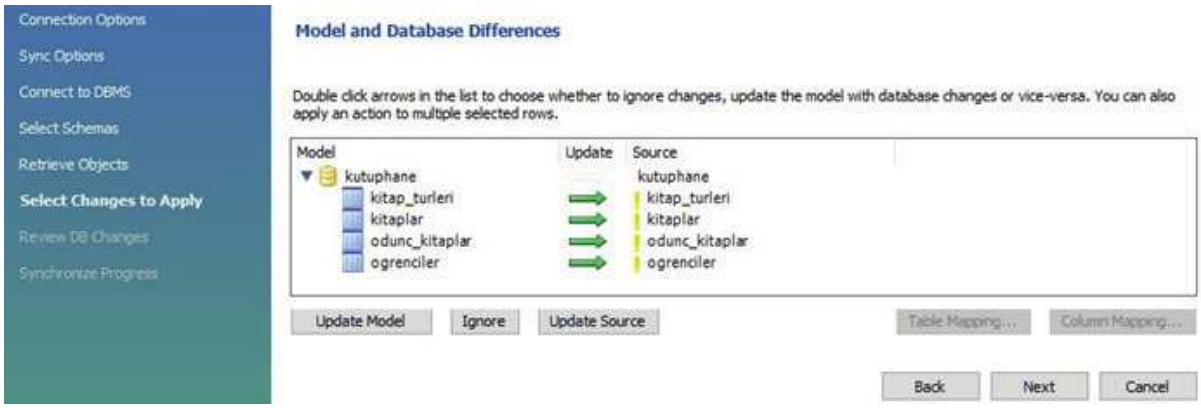
Görsel 6.77: İlişkileri veri tabanına uygulama-Synchronize Model

Görsel 6.78'de ilişkilerin uygulanacağı veri tabanı seçilir ve "Next" düğmesi tıklanarak işleme devam edilir.



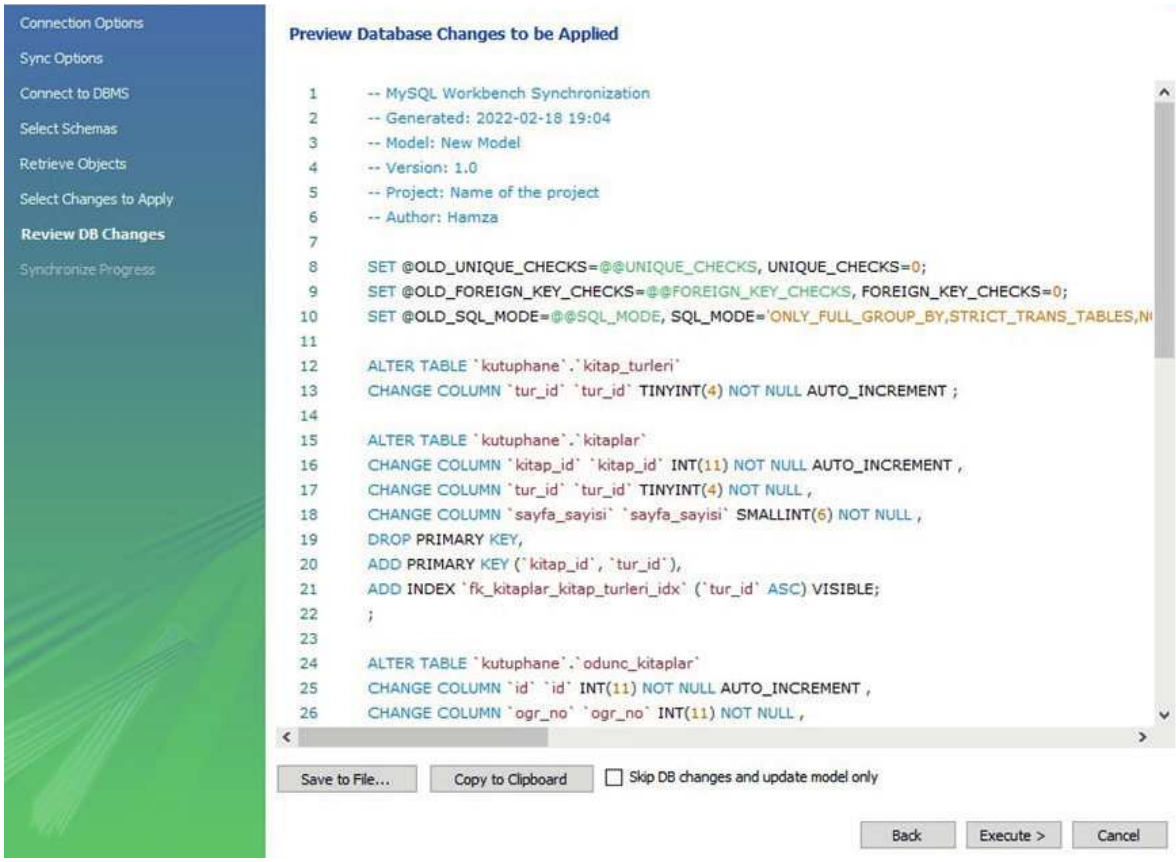
Görsel 6.78: İlişkilerin uygulanacağı veri tabanı seçimi

Görsel 6.79'daki ekranda ilişkilerin uygulanacağı tablolar gösterilir.



Görsel 6.79: İlişkilerin uygulanacağı tabloların seçimi

Görsel 6.80'deki ekranda ilişkilerin oluşturulması için çalıştırılacak SQL komutları görülür. "Execute" düğmesine tıklanarak tablolar arası ilişkiler oluşturulur.



Görsel 6.80: İlişkiyi oluşturacak olan SQL komutları

Sıra Sizde

Hastanelerdeki randevu alma işlemlerinin kaydedilmesi için bir veri tabanı tasarımı yapınız. Bu veri tabanı; hasta, doktor, bölüm ve randevu gibi bilgilerin kaydedileceği şekilde tasarlanmalıdır. Ayrıca tablolar arasında ilişkiler oluşturulmalı, gerekirse alt tablolar oluşturularak veri tekrarı önlenmelidir.

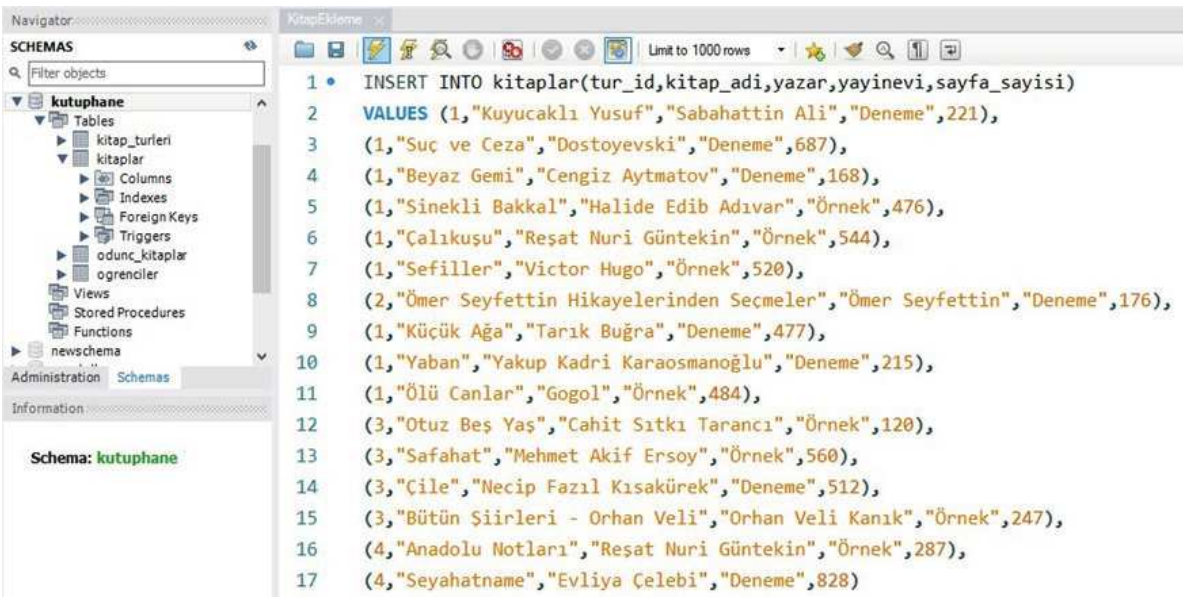
6.5.3. İlişkisel Veri Tabanı Tablolarına Veri Girişi Yapılması

İlişkisel veri tabanında sorgulama işlemlerinin yapılabilmesi için tablolarda verilerin bulunması gerekir. Bundan dolayı tablolara veri girişi yapılır. Veri girişi yapılırken PK alanların tekrar etmemesine ve FK alanların da bağlı olduğu tablodaki değerlerden birini almasına dikkat edilmelidir.

Görsel 6.81'de veri tabanına kitap türlerini kaydeden SQL sorgusu görülür. İlk olarak "kitap_turleri" tablosuna veri eklenmesinin nedeni, "kitap" tablosunun bu verileri kullanacak olmasıdır. **Tur_id** alanı otomatik artan olarak belirtildiği için o alana veri girişi yapılmamıştır. Sorgu çalıştırıldıktan sonra tablodaki veriler listelendiğinde verilerin kaydedildiği görülür.

Görsel 6.81: kitap_turleri tablosuna veri ekleme

Görsel 6.82'deki sorgu çalıştırıldığında kitaplar tablosuna 16 adet veri eklendiği görülür. **kitap_id** alanı otomatik artan olarak belirtildiği için o alana veri girişi yapılmamıştır. Bu alandaki değer otomatik olarak artacaktır. Ayrıca iki tablo arasında bire çok ilişki yapıldığı için **tur_id** alanına yalnızca "kitap_turleri" tablosunda bulunan tür değerleri girilmiştir. Örneğin **tur_id** değerine roman için 1, şiir için 3 değeri girilmiştir. Bu değerler, kitap_turleri tablosunda bu türlere karşılık gelen id değerleridir.

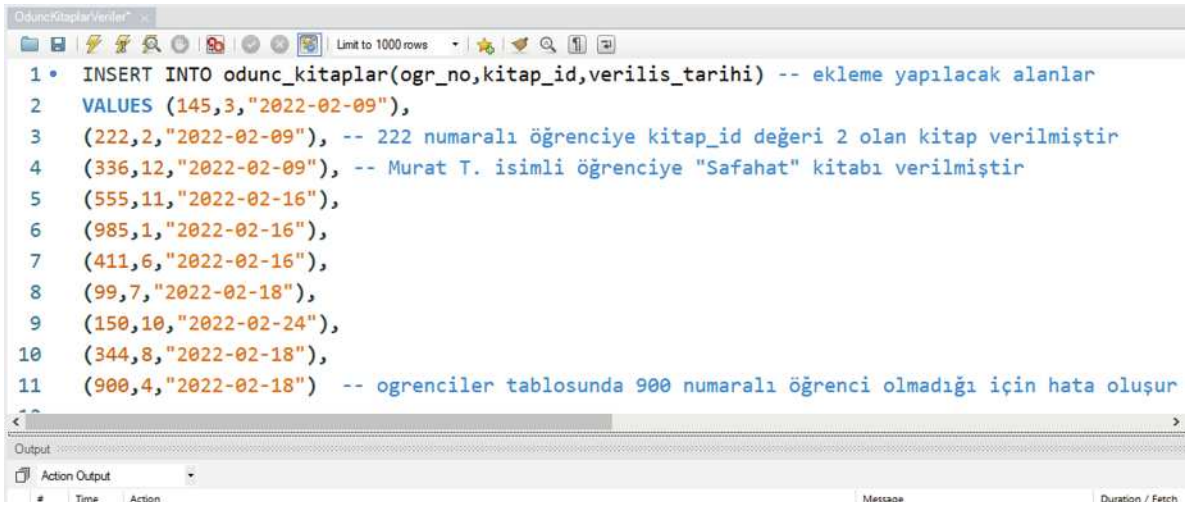


Görsel 6.82: Kitaplar tablosuna veri ekleme

Görsel 6.83'teki sorgu çalıştırıldığında öğrenciler tablosuna 13 adet kayıt eklendiği görülür. "ogrenci_no" alanı PK olarak ayarlandığı için bu alana tekrar eden veri girilmemelidir.

Görsel 6.83: Ogrenciler tablosuna veri ekleme

Görsel 6.84'teki SQL sorgusunda "odunc_kitaplar" tablosuna veri eklenirken karşılaşılan hata mesajı gösterilmiştir. Bu hatanın sebebi, **ogr_no** alanının bağlı olduğu "ogrenciler" tablosunda 900 numaralı öğrenci kaydının bulunmamasıdır. Son kayıt silinerek sorgu çalıştırıldığında verilerin tabloya eklendiği görülür. **teslim_tarihi** ve **aciklama** alanları boş bırakılabilir olarak ayarlandığı için bu alanlara veri girişi yapılmadığında hata oluşmaz.



Görsel 6.84: Odunc_kitaplar tablosuna veri ekleme

SQL sorgularına yorum satırı eklemek için "--" simgesi kullanılır ve yanında boşluk bırakılarak yorum yazılır.

Not Tablolara veri girişi yapılırken otomatik artan sayı şeklinde ayarlanan id değerleri görsellerdeki ile bire bir aynı olmayabilir. Tablolara kayıt yapılmaya çalışıldığında veya veriler silindiğinde sistem o id değerini diğer kayıtlara vermez ve sonraki sayıdan devam eder. Örneğin bir tabloya 20 adet yanlış kayıt yapılmaya çalışıldıysa eklenecek doğru kayıt, 21'den başlayarak devam eder. Bu durum, herhangi bir sorun oluşturmaz. Bağlantılı tablolara veri girişi yapılırken o id değerine göre giriş yapılmalıdır.

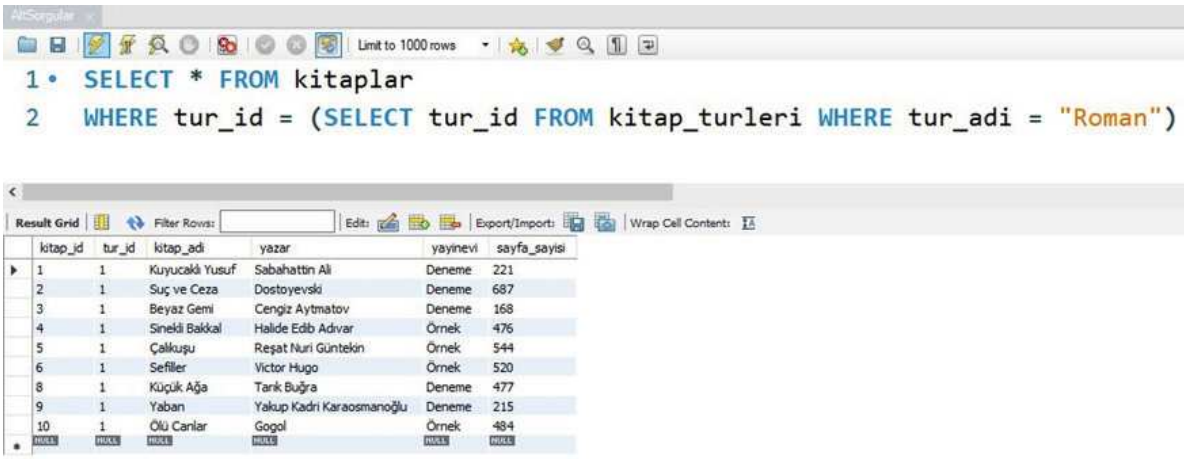
6.5.4. İlişkisel Veri Tabanında Sorgular

İlişkisel veri tabanlarında sorgulama yapılması için birden fazla tablonun aynı sorgu içinde kullanılması gerekir. Tablolar tek tek listelendiği zaman genellikle kullanıcılar için anlamsız görünen veriler ortaya çıkar. Bu verilerin anlamlı hâle gelmesi için tabloların belirli alanlarının birleştirilerek listelenmesi gerekir.

Alt Sorgular

Alt sorgular, bir sorgudan elde edilen sonucun başka bir sorguda kullanılması ile oluşturulan sorgulardır. Alt sorgu denilen içerideki sorgunun sonucu, ana sorguda koşul belirtilmesi için kullanılır.

Görsel 6.85'teki sorguda kitaplar tablosundaki kitap türü "Roman" olan kitaplar listelenmiştir. Kitaplar tablosunda kitap türünün **tur_id** değeri tutulduğu için bu sorgulama işleminde "kitap_turleri" tablosuna da ihtiyaç vardır. Buradaki alt sorguda "Roman" türünün **tur_id** değeri döndürülmüştür. İlk olarak alt sorgunun çalıştığı düşünülürse sorgu, "SELECT * FROM kitaplar WHERE tur_id = 1" hâlini almıştır.

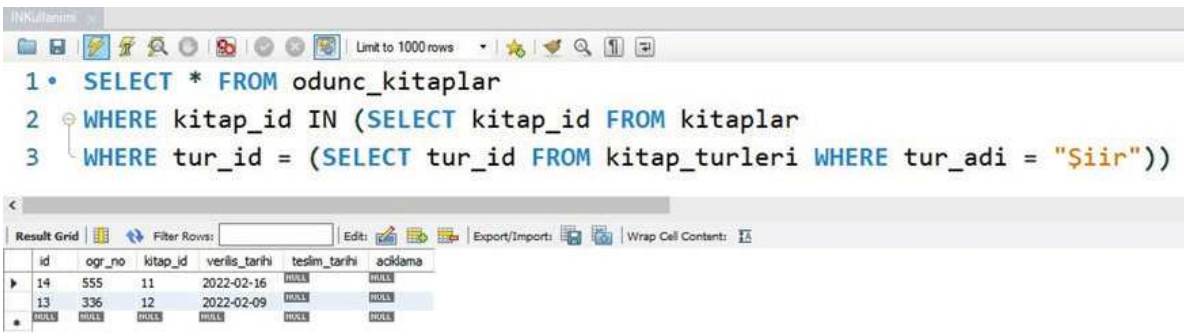


Görsel 6.85: Alt sorgu kullanımı

IN Kullanımı

Alt sorgu kayıtlarının tümüyle eşleşen ana sorgu kayıtlarını almak için **IN** anahtar sözcüğü kullanılır. IN, çoklu satır döndüren alt sorgularda kullanılır. Aranan değerin o kayıtların içinde olup olmadığı kontrol edilir.

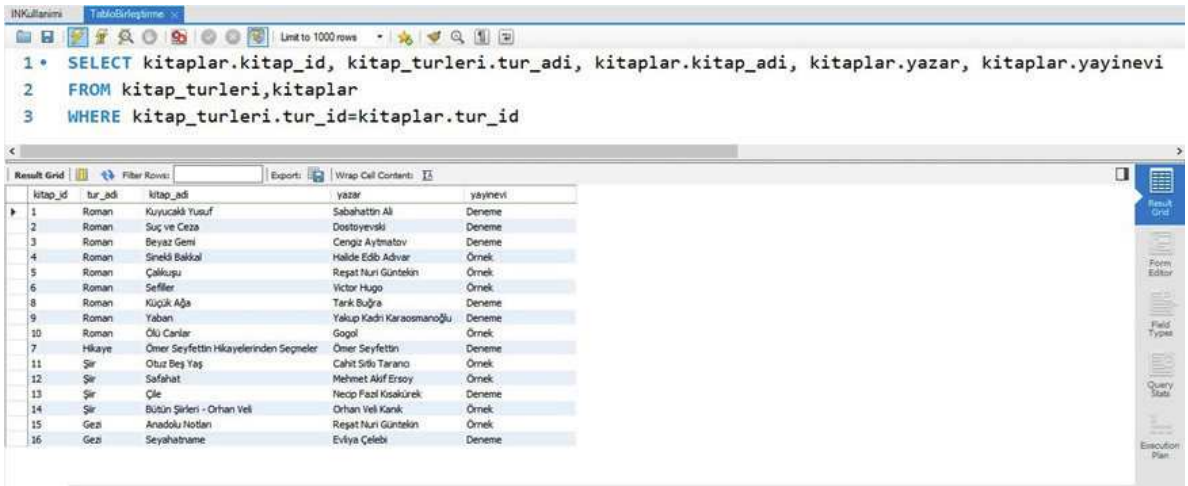
Görsel 6.81'deki sorguda "odunc _kitaplar" tablosundaki kayıtlardan "Şiir" türünde ödünç kitap alan öğrencileri listeleyen SQL sorgusu yazılmıştır. Kitaplar tablosundan bu türe ait birden fazla kitap listelenebileceği için "IN" anahtar sözcüğü kullanılmıştır.



Görsel 6.86: Alt sorgularda "IN" kullanımı

Tablo Birleştirme

SQL ile iki veya daha fazla tablodan aynı anda veri çekerek kullanıcıların verileri daha anlamlı görüntülemesi mümkündür. "Kitaplar" tablosu ile "kitap_turleri" tablosu birleştirilerek **tur_id** değeri yerine **tur_adi** değerinin yazılması sağlanmıştır (Görsel 6.87). Birleştirme işlemi, tablolardaki PK ve bağlı olduğu FK alanları koşul kısmında birbirine eşitlenerek yapılır. Bu sorgu, tablolara ve alanlara takma isim verilerek de kullanılabilir.

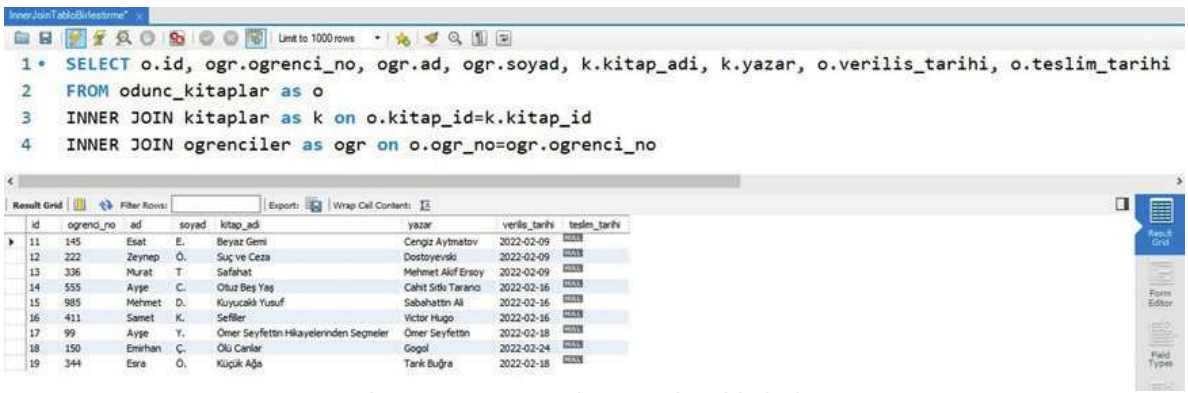


Görsel 6.87: Tablo birleştirme

INNER JOİN Kullanarak Tablo Birleştirme

Tabloları birleştirmede en çok tercih edilen yöntemlerden biri de **join** (birleştirici) kullanımıdır. Bu yöntemle tablolar, ortak sütunlar yardımı ile birbirine bağlanarak istenen alanlar listelenebilir. Birleştirme işlemi INNER JOIN sözcüğünden sonra birleştirilecek tablonun adı yazılarak yapılır. "on" anahtar kelimesinden sonra ise tabloların hangi alan üzerinden birleştirileceği belirtilir. Bunlar, tablolardaki FK ve buna bağlı olan PK alanlardır. Ayrıca "as" anahtar kelimesi sayesinde tablolar takma isimlerle kullanılır.

Görsel 6.88'deki SQL sorgusunda "odunc_kitaplar", "kitaplar" ve "öğrenciler" tabloları birleştirilerek istenen alanların listelenmesi sağlanmıştır. Bu sorgudan dönen liste incelendiğinde ödünç kitap verme işlemi ile ilgili bilgilerin daha anlamlı hâle geldiği görülür.



Görsel 6.88: INNER JOIN komutuyla tablo birleştirme

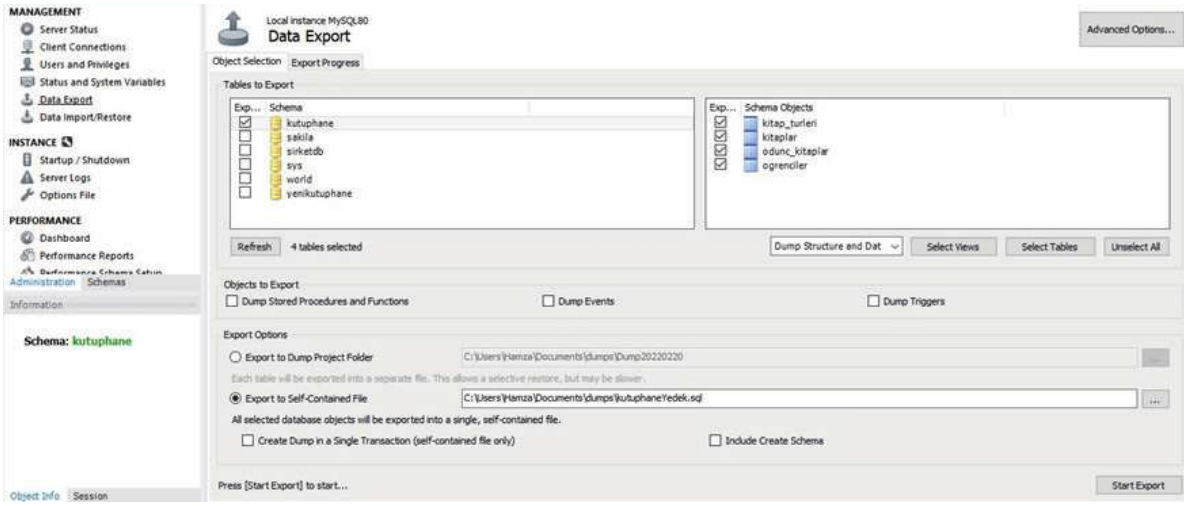
Sıra Sizde

"Kitaplar" ve "kitap_turleri" tablolarını INNER JOIN işlemi ile birleştirerek kitap id, tür adı, kitap adı, yazarı ve sayfa sayısı bilgilerini listeleyen SQL sorgusunu yazınız.

6.6. MySQL VERİ TABANININ YEDEĞİNİ ALMA VE GERİ YÜKLEME

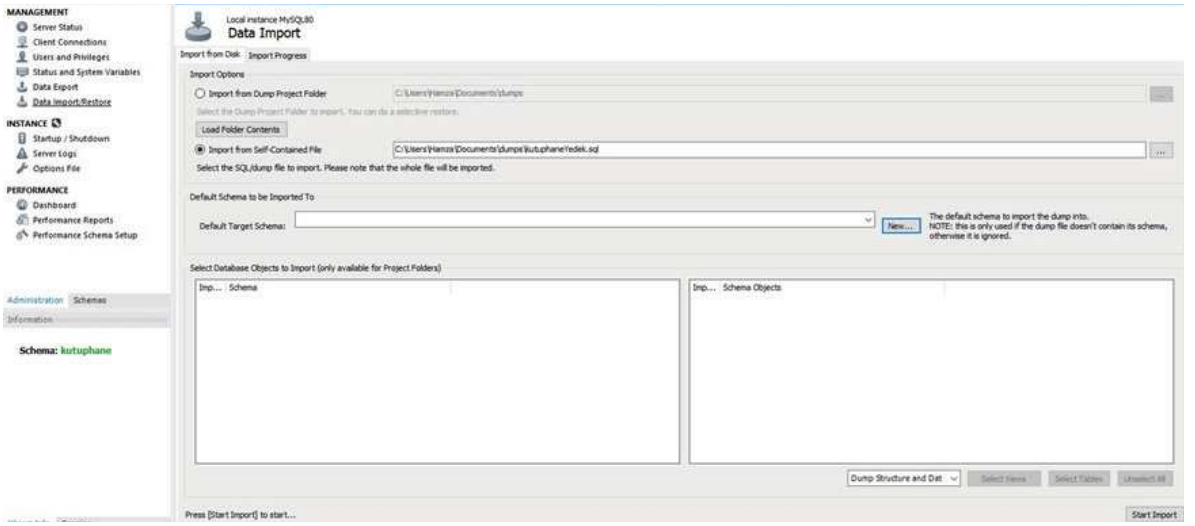
Veri tabanı ile çalışılırken bazı durumlarda veri tabanının yedeğinin alınması veya taşınması gerekebilir. Bu durum, veri tabanının başka bilgisayara taşınıp orada çalışılması veya proje tamamlandıktan sonra sunucuya yüklenmesi için olabilir. Bu nedenle veri tabanının yerel sunucudan alınabilmesi, başka bir bilgisayar veya sunucuya yüklenebilmesi çok önemlidir.

Veri tabanının yerel sunucudan alınması için ilk olarak Görsel 6.89'da görüldüğü gibi **Navigator** penceresinde **Administration** sekmesine geçilir ve **Data Export** seçeneğine tıklanır. "Tables to Export" bölümünden dışa aktarılmak istenen veri tabanı seçilir. Ardından "Export to Self-Contained File" seçeneği işaretlenerek dosya yolu ve ismi belirtilir. Son olarak da sağ alt bölümdeki **Start Export** düğmesi tıklanarak işlem başlatılır. Aksi belirtilmediyse dosyanın Belgelerim/dumps klasörü içinde olduğu görülür.



Görsel 6.89: Data Export işlemiyle yedek alma

Yedeği alınan veya Export işlemi yapılan bir veri tabanının sunucuya dâhil edilmesi için **Administration** sekmesinden **Data Import/Restore** seçeneğine tıklanır. Öncelikle "Import From Self-Contained File" seçeneği işaretlenerek içe aktarılacak dosya seçilir. Daha sonra "Default Target Schema" bölümünden yedeğin hangi veri tabanı içine aktarılacağı seçilir. İçeri aktarılacak dosya için bir veri tabanı oluşturulmadıysa **New** düğmesi tıklanarak yeni bir veri tabanı oluşturulur. Son olarak da sağ alt bölümdeki **Start Import** düğmesine tıklanarak veri tabanı içe aktarılır (Görsel 6.90).



Görsel 6.90: Data Import/Restore işlemiyle alınan yedeği geri yükleme

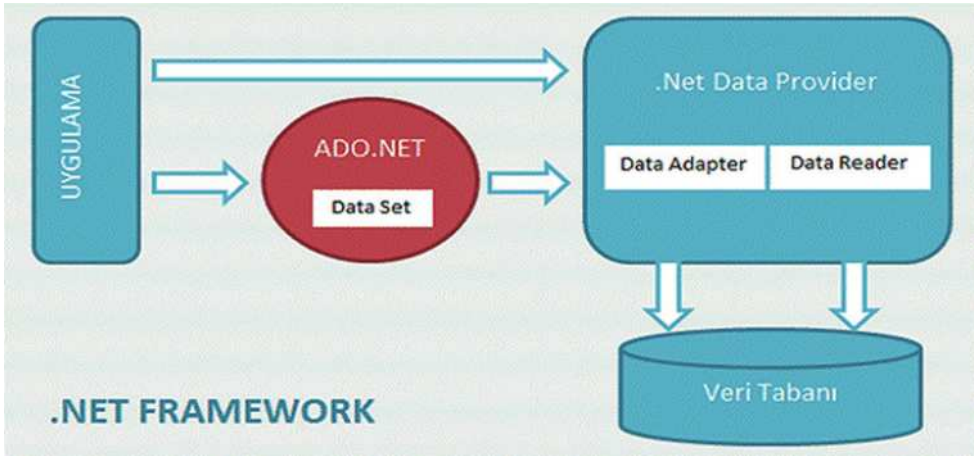
6.7. NESNE TABANLI PROGRAMLAMADA VERİ TABANI KULLANIMI

Veri tabanı, geliştiriciler tarafından kolay bir şekilde kullanılsa da kullanıcıların veri tabanını kullanabilmesi için bir uygulama arayüzü (form uygulamaları, web siteleri, mobil uygulamalar) oluşturulmalıdır. Kullanıcıların veri tabanı üzerinde sorgu yazması veya nesne tabanlı programlama üzerinde kod yazması beklenemez. Bu yüzden veri tabanındaki bilgiler nesne tabanlı programlama üzerinde oluşturulan arayüzlere eklenerek kullanıcının veri tabanı işlemlerini bu arayüz üzerinden kolaylıkla yapabilmesi sağlanır.

6.7.1. ADO.NET (ActiveX Data Objects.NET)

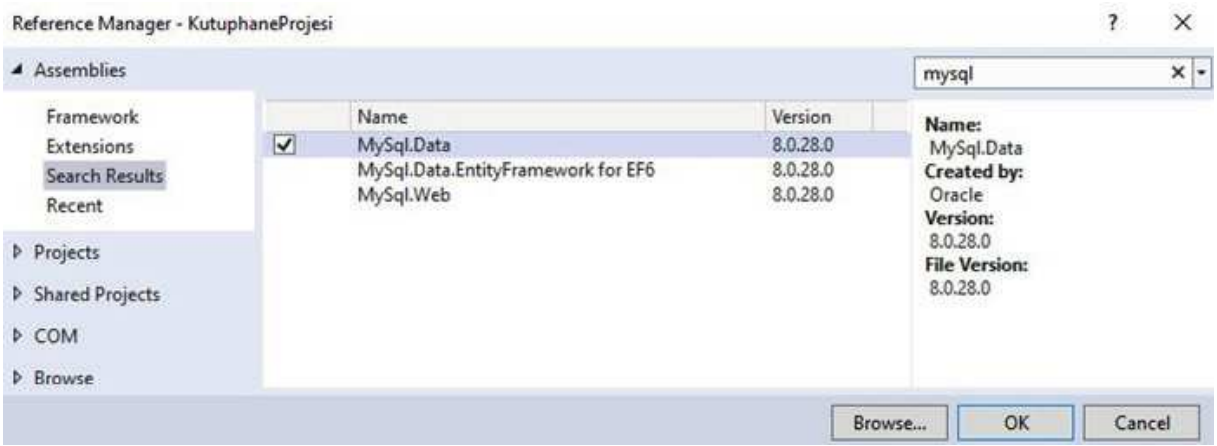
ADO.NET, ActiveX tabanlı çalışan bir veri erişim teknolojisidir. Uygulama ile veri tabanı arasında köprü görevi görerek uygulama üzerinde yazılan SQL sorgularının (ekleme, silme, güncelleme ve listeleme) veri tabanı üzerinde çalıştırılmasını sağlar.

ADO.NET ile MS SQL, MySQL, Oracle, PostgreSQL ve Access gibi birçok veri tabanı **.NET** platformunda kolaylıkla kullanılabilir.



Görsel 6.91: ADO.NET şeması

ADO.NET teknolojisi ile MySQL veri tabanı işlemlerinin yapılabilmesi için öncelikle MySql.Data kütüphanesinin, referanslar bölümüne eklenmesi gerekir. Bunun için "Solution Explorer" penceresindeki "References" klasörüne sağ tıklanarak "Add Reference" seçeneği seçilir. Gelen pencerede arama bölümüne mysql yazılarak **MySql.Data** seçilir ve "OK" butonuna tıklanır (Görsel 6.92).



Görsel 6.92: MySql.Data kütüphanesinin projeye eklenmesi

Veri tabanı bağlantısının sağlanabilmesi için öncelikle işlem yapılacak sayfanın üst bölümündeki "System.Data" ve "MySql.Data.MySqlClient" kütüphanelerinin eklenmesi gerekir. Daha sonra bu kütüphanelerde bulunan sınıflar kullanılarak veri tabanı işlemleri gerçekleştirilir. Bu sınıflar şunlardır:

1. **MySqlConnection:** Veri tabanı ile bağlantının kurulması için kullanılan sınıftır. Parametre olarak aldığı "connectionString" değerine göre belirtilen veri tabanına bağlantı sağlar. Ayrıca "Open" ve "Close" metotları kullanılarak oluşturulan bağlantıyı açmak ve kapatmak mümkündür.

- **connectionString:** MySQL veri tabanına bağlanmak için gerekli olan bağlantı cümlesidir. Hangi sunucudaki hangi veri tabanına bağlanılacağı burada belirtilir.

```
MySqlConnection baglanti = new MySqlConnection("Server=localhost;Database=kutuphane;Uid=root;Pwd=19675561");
```

Görsel 6.93: My SQL veri tabanı bağlantısı

Server, bağlantı sağlanacak sunucunun belirtildiği bölümdür.

Database, kullanılacak olan veri tabanının belirtildiği bölümdür.

Uid, veri tabanı sunucusuna bağlanmak için tanımlanan kullanıcı adının yazılacağı bölümdür.

Pwd, veri tabanı sunucusuna bağlanmak için oluşturulan şifrenin yazılacağı bölümdür.

Not Bağlantı cümlesi yazılırken dikkat edilmelidir. Yapılacak en küçük hata, veri tabanı bağlantısının başarısız olmasına yol açar.

2. **MySqlCommand:** MySQL veri tabanı üzerinde sorgunun çalıştırılması için kullanılan sınıftır. Veri tabanı bağlantısı sağlandıktan sonra bu sınıf ile veri tabanı üzerinde sorgu çalıştırılabilir. MySqlCommand sınıfı kullanılırken sorgu cümlesi (CommandText) ve bağlantı (Connection) parametreleri hatasız bir şekilde belirtilmelidir (Görsel 6.94).

- **ExecuteNonQuery:** MySqlCommand nesnesinde belirtilen komutun çalıştırılması için kullanılır. Bu metot kullanılmadan önce veri tabanı bağlantısının açık olması gerekir. Genellikle ekleme, silme ve güncelleme sorgularında kullanılır. Etkilenen kayıt sayısı değerini geri döndürür.

```
MySqlConnection baglanti = new MySqlConnection("Server=localhost;Database=kutuphane;Uid=root;Pwd=19675561");
MySqlCommand komut = new MySqlCommand("INSERT INTO kitap_turleri(tur_adi) values('Bilgisayar')", baglanti);
baglanti.Open();
komut.ExecuteNonQuery();
baglanti.Close();
```

Görsel 6.94: MySqlCommand sınıfının kullanımı

- **ExecuteScalar:** Çalıştırılan sorgu sonucunda tek bir değer alınacaksa bu metot kullanılır. Örneğin SUM metoduyla toplam değer listelenmek istenirse sorgu bu yöntemle çalıştırılarak toplam değer, bir değişkene aktarılabilir.

- **ExecuteReader:** MySqlCommand nesnesinde belirtilen sorgu sonucunda dönen kayıtların listelenmesi için kullanılır. Kayıtların tek tek okunması ve ona göre bir işlem yapılması istenirse sorgunun bu metot ile çalıştırılması gerekir.

3. **MySqlDataReader:** MySqlCommand nesnesinin ExecuteReader metodu ile çalıştırdığı listeleme sorgusundan gelen verilerin okunması için kullanılır. Bütün kayıtlar satırlar hâlinde tek tek okunacağı için az miktardaki verilerin okunmasında kullanılır.

4. **MySqlDataAdapter:** Bir sorgu sonucunda listelenen verilerin DataSet, DataTable, DataView gibi tablo hâlinde veri tutma özelliği olan nesnelere aktarılması için kullanılır. Parametre olarak SQL sorgusu ve bağlantı nesnesi veya doğrudan MySqlCommand nesnesi belirtilir. Verilen bağlantı üzerinde sorguyu çalıştırarak dönen sonucu "Fill" metodunda belirtilen nesnenin içine aktarır.

Görsel 6.95'te görülen MySqlDataAdapter nesnesinin kullanımında, bağlantı üzerinde SELECT sorgusu çalıştırılarak sonuç DataTable nesnesine aktarılır. Ayrıca MySqlCommand nesnesinden farklı şekilde bağlantısız olarak da çalışabilir. Bu durumda bağlantı açıp kapatma işlemini MySqlDataAdapter nesnesi otomatik olarak yapacaktır.

```
string baglantiCumlesi = "Server=localhost;Database=kutuphane;Uid=root;Pwd=19675561";
MySqlConnection baglanti = new MySqlConnection(baglantiCumlesi);
sorguCumlesi = "SELECT * FROM kitap_turleri";
MySqlDataAdapter adapter = new MySqlDataAdapter(sorguCumlesi, baglanti);
DataTable tablo = new DataTable();
adapter.Fill(tablo);
grdKitapTurleri.DataSource = tablo;
```

Görsel 6.95: MySqlDataAdapter nesnesiyle alınan verilerin DataGridView'de gösterilmesi

MySqlDataAdapter sınıfı yalnızca verinin listelenmesi için kullanılmaz. Gösterilen kullanımın dışında listelemek için SelectCommand, eklemek için InsertCommand, güncellemek için UpdateCommand ve silmek için de DeleteCommand özellikleri ile bu sorgular için de kullanılabilir.

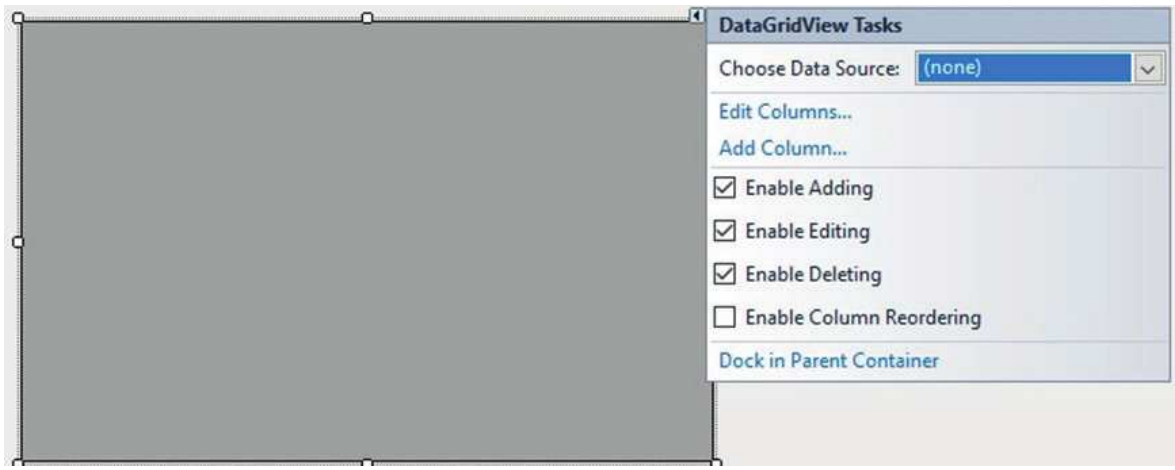
5. **DataTable:** Verileri tablo hâlinde saklayan nesnelerdir. Satır (DataRow) ve sütunlardan (DataColumn) oluşur. Sorgu sonucunda listelenen veriler DataTable nesnesine doldurularak program içinde kullanılabilir. Tablo hâlinde veri tutan bir değişken olarak da düşünülebilir.

6. **DataSet:** İçinde bir veya birden fazla DataTable barındıran nesnelerdir. İçindeki DataTable nesneleri ile tablolar hâlinde veri tutar. Yalnızca bir tablo kullanılacaksa DataTable nesnesini kullanmak daha doğru olur. DataSet'in en büyük avantajı, bağlantısız olarak çalışabilmesidir. Veri tabanı tabloları bu nesne içine aktarıldıktan sonra bağlantı kesilebilir. Bu noktadan sonra DataSet program içinde bir veri tabanı gibi kullanılabilir. Tablo ekleme ve silme, veri ekleme, silme, güncelleme ve listeleme gibi işlemler bu nesne üzerinde yapılarak son hâli veri tabanına gönderilebilir.

6.7.2. DataGridView Bileşeni

DataGridView bileşeni, veri tabanından alınan verilerin uygulama üzerinde gösterilmesini sağlayan bir form elemanıdır. Veri tabanında olduğu gibi satır ve sütunlardan oluşan tablo gösterimini sağlar. "DataSource" özelliği, bir veri kaynağına bağlanarak kullanılır. Bu veri kaynağının tablo hâlinde veri tutan bir nesne olması gerekir.

Görsel 6.96'da forma eklenen bir DataGridView nesnesi görülür.



Görsel 6.96: DataGridView görünümü

DataGridView bileşeninin bazı özellikleri şunlardır:

DataSource: Nesnenin veri kaynağının (hangi verilerin listeleneceğinin) belirlendiği özelliktir.

AllowUserToAddRows: Bu özellik en altta kullanıcının veri eklemesi için boş bir satır oluşturur.

EditMode: "EditProgrammatically" olarak ayarlanırsa yalnızca programlama yöntemi ile nesne üzerinde değişiklik yapılabilir. Diğer seçeneklerde ise nasıl bir değişiklik istenirse ona göre seçim yapılır.

SelectionMode: Datagridview hücrelerine tıklandığında nasıl seçim yapılacağı belirtilir. "FullRowSelect" seçeneği seçilirse tıklanan hücrenin bulunduğu satırın tamamı seçili hâle gelir.

MultiSelect: Aynı anda birden fazla satırın seçilip seçilmeyeceği burada belirlenir.

Columns["sütun adı"].HeaderText: Listelenen sütunun başlığının ne olacağını belirtmek için kullanılır. Değer belirtilmezse veri tabanındaki alan adı görüntülenir.

Columns["sütun adı"].Width: Belirtilen sütunun genişliğinin ne kadar olacağı ayarlanır.

AutoSizeColumnsMode: Sütun genişliklerinin otomatik olarak ayarlanması için kullanılır. "Fill" seçeneği seçilirse tablo sütunları DataGridView nesnesinin tamamını kaplar.

CurrentRow.Cells["sütun adı"].Value: Seçili satırdaki hücrenin içindeki veriyi almak için kullanılır.

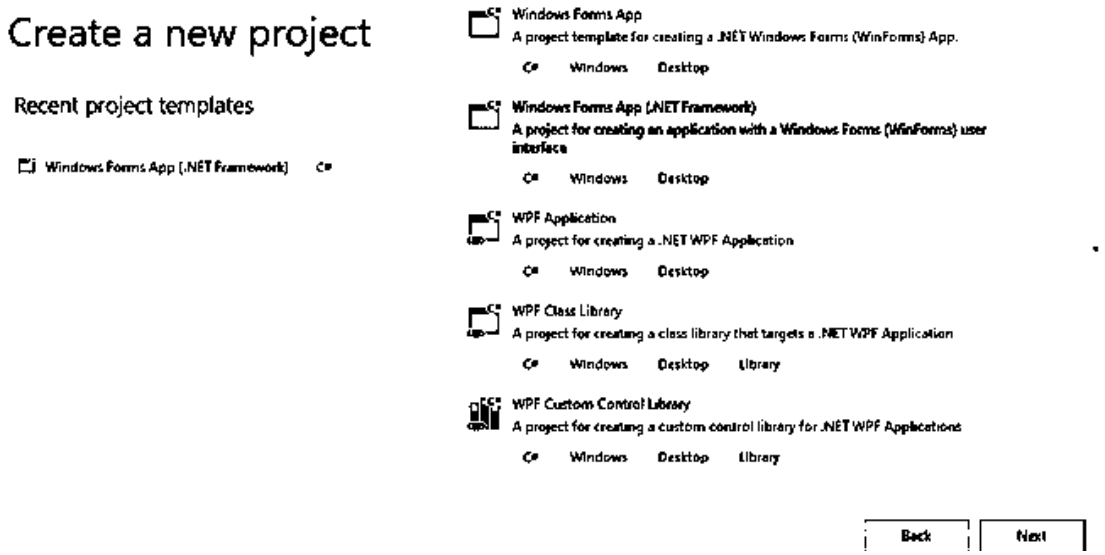
6.8. KÜTÜPHANE OTOMASYONU PROJESİNİN GELİŞTİRİLMESİ

Bu bölümde daha önce oluşturulan "kutuphane" veri tabanı kullanılarak **Kütüphane Otomasyonu Projesi** yapılacaktır. Bu projede kullanıcıların öğrenci ve kitaplarla ilgili ekleme, silme, güncelleme ve listeleme işlemlerinin yanı sıra ödünç kitap verme ve geri alma işlemlerini de program arayüzünden yapmaları sağlanacaktır.

Not Proje geliştirilirken form elemanları için farklı isimlendirmeler kullanılabilir fakat proje kodlarında da bu isim değişikliklerinin yapılması gerekir. Aksi hâlde program hata verir. Ayrıca program görsel olarak zenginleştirilebilir ve programa başka özellikler eklenebilir.

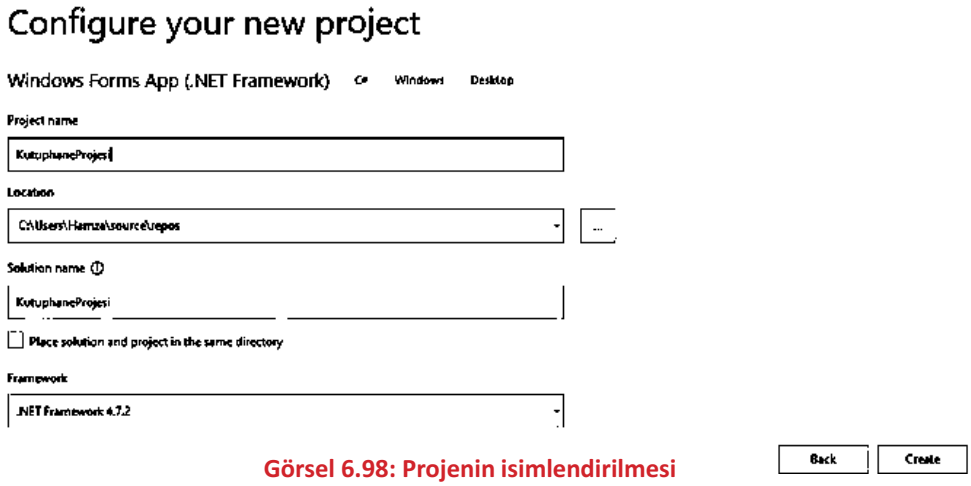
6.8.1. Windows Form Projesinin Oluşturulması

Kod editörü programı çalıştırıldıktan sonra yeni bir Windows form projesi oluşturulur (Görsel 6.97).



Görsel 6.97: Kod editöründe form uygulaması oluşturma

Görsel 6.98'de görüldüğü gibi **Project name** bölümünden projenin ismi "KutuphaneProjesi" olarak belirlenir. **Create** butonuna tıklanarak projenin oluşturulması sağlanır.



Görsel 6.98: Projenin isimlendirilmesi

6.8.2. Veri Tabanı Bağlantı Sınıfının Oluşturulması

Projede ilk olarak veri tabanı bağlantısının yapılacağı bir sınıf oluşturulur. Bağlantı cümlesinin her sayfada tekrar yazılması yerine bir sınıf içinde bağlantı oluşturulup diğer sayfalarda bu sınıftan çağrılması kullanım açısından bir kolaylıktır.

Öncelikle **App.config** dosyasının içinde **<connectionStrings> </connectionStrings>** düğümlerinin arasında bağlantı cümlesi tanımlanır. Veri tabanı ile ilgili herhangi bir değişiklik durumunda yalnızca bu ayar dosyasının içindeki connectionString değerinde değişiklik yapılması yeterlidir.

Bağlantı sınıfının oluşturulması için projeye sağ tıklanarak Add>Class seçeneği seçilir. Bağlantı sınıfı, "VeriTabaniIslemleri.cs" şeklinde isimlendirilerek projeye eklenir. App.config dosyasının içine eklenen **connectionString**'in kullanılabilmesi için "System.Configuration" kütüphanesi sayfaya dâhil edilir. Daha sonra Görsel 6.99'daki gibi veri tabanı bağlantı komutları yazılır.

App.config*

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.8"/>
    </startup>
    <connectionStrings>
        <add name="kutuphaneBaglantiCumlesi"
             connectionString="Server=localhost;Database=kutuphane;Uid=root;Pwd=19675561;"/>
    </connectionStrings>
</configuration>
```

VeriTabaniIslemleri.cs*

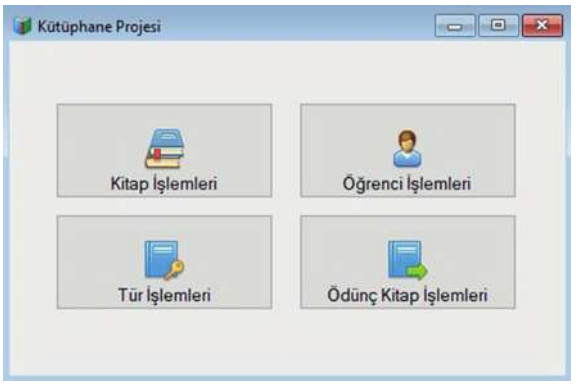
KutuphaneProjesi | KutuphaneProjesi.VeriTabaniIslemleri | baglantiCumlesi

```
using System.Configuration;
namespace KutuphaneProjesi
{
    8 references
    internal class VeriTabaniIslemleri
    {
        string baglantiCumlesi = ConfigurationManager.ConnectionStrings["kutuphaneBaglantiCumlesi"].ConnectionString;
        // App.config dosyasının içindeki veri tabanı bağlantı cümlesi bir değişkene aktarıldı
        5 references
        public MySqlConnection baglan()
        {
            MySqlConnection baglanti = new MySqlConnection(baglantiCumlesi);
            // Veri tabanı bağlantısı oluşturuldu
            MySqlConnection.ClearPool(baglanti); // Önceki bağlantılar temizlendi
            return baglanti; // Oluşturulan bağlantı fonksiyonun çağırıldığı yere gönderildi
        }
    }
}
```

Görsel 6.99: Veri tabanı bağlantı sınıfının hazırlanması

6.8.3. Proje Ana Sayfasının Hazırlanması

Projede yönlendirme işlemlerinin yapılması için ilk olarak bir Ana Sayfa hazırlanır. Bu sayfa içinde diğer sayfaların açılmasını sağlayacak butonlar oluşturulur. Tasarımın güzel görünmesi için butonlara görseller eklenir. Ana Sayfa Görsel 6.100'de görüldüğü gibi oluşturulur. Form nesnesinde ve içindeki form elemanlarında bazı özelliklerin ayarlanması gerekir.



Görsel 6.100: Proje Ana Sayfa formunun oluşturulması

Form Nesnesi

"Text" özelliği, **Kütüphane Projesi** olarak belirlenir.

"Name" özelliği, **formAnaSayfa** olarak belirlenir.

"Size" özelliği, **450;300** olarak belirlenir.

"FormBorderStyle" özelliği, form ekranının büyütülüp küçültülmesi kötü görünmesine yol açacağı için **FixedToolWindow** olarak belirlenerek formun boyutlandırılması engellenir.

"StartPosition" özelliği, **CenterScreen** olarak belirlenir. Form, ekranın tam ortasında açılır.

Button Nesneleri

"Text" özellikleri; **Kitap İşlemleri, Öğrenci İşlemleri, Tür İşlemleri** ve **Ödünç Kitap İşlemleri** olarak Görsel 6.110'de görüldüğü gibi belirlenir.

"Name" özellikleri; **btnKitap, btnOgrenci, btnTur, btnOdunc** olarak belirlenir.

"Image" özellikleri ile butonlara ilgili görseller eklenir. Görsellerin bire bir aynı olması gerekmez.

"Size" özellikleri, **170;75** olarak belirlenir.

Ana Sayfa Kodlarının Yazılması

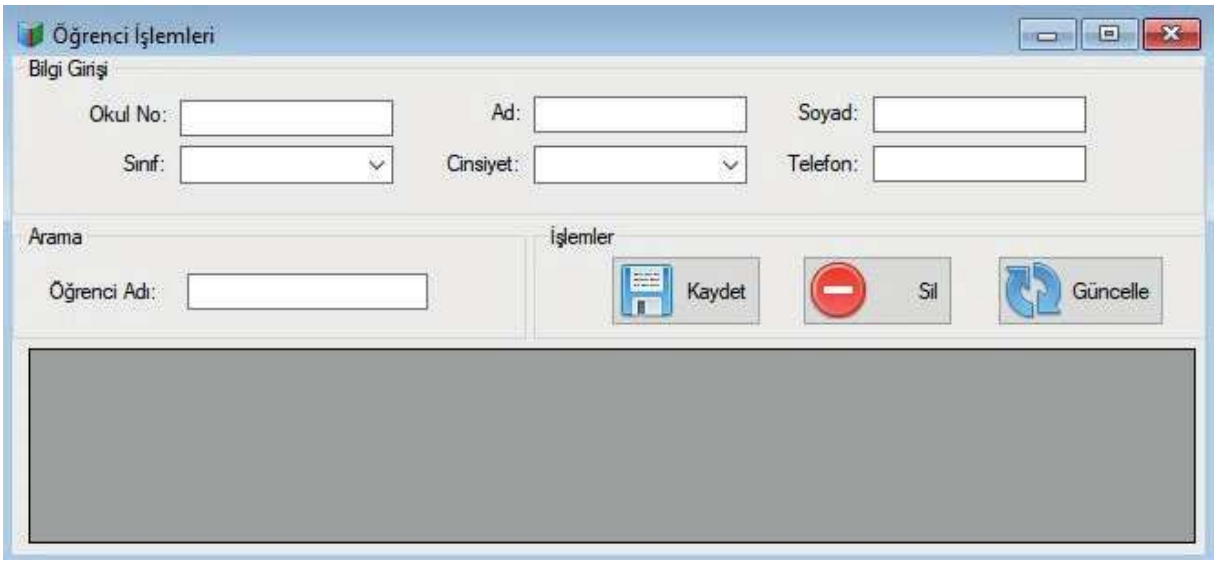
Oluşturulan Ana Sayfa bölümünde butonlara tıklandığında ilgili sayfalara yönlendirme işleminin yapılması için bazı kodların yazılması gerekir. Bu kodlar, butonların "Click" olayına yazılmalıdır (Görsel 6.101).

```
private void btnKitap_Click(object sender, EventArgs e)
{
    formKitap kitap = new formKitap();
    kitap.ShowDialog();
}
1 reference
private void btnOgrenci_Click(object sender, EventArgs e)
{
    formOgrenci ogrenci = new formOgrenci();
    ogrenci.ShowDialog();
}
1 reference
private void btnTur_Click(object sender, EventArgs e)
{
    formKitapTur kitapTur = new formKitapTur();
    kitapTur.ShowDialog();
}
1 reference
private void btnOdunc_Click(object sender, EventArgs e)
{
    formOduncKitap oduncKitap = new formOduncKitap();
    oduncKitap.ShowDialog();
}
```

Görsel 6.101: Ana Sayfa kodları

6.8.4. Öğrenci İşlemleri Sayfasının Hazırlanması

Öğrenci İşlemleri sayfası; öğrenciler ile ilgili ekleme, silme, güncelleme, listeleme ve arama gibi işlemlerin yapılacağı sayfadır. Yeni bir form oluşturularak formOgrenci adı verilir. Ardından Görsel 6.102'deki form tasarımı yapılır. Formun alt bölümündeki "DataGridView" nesnesi, verilerin listelenmesi için kullanılır.



Görsel 6.102: Öğrenci İşlemleri form tasarımı

Form nesnesinde ve içindeki form elemanlarında bazı özelliklerin ayarlanması gerekir.

Form Nesnesi

"Text" özelliği, **Öğrenci İşlemleri** olarak belirlenir.

"Name" özelliği, **formOgrenci** olarak belirlenir.

"Size" özelliği, **670;550** olarak belirlenir.

"FormBorderStyle" özelliği, form ekranının büyütülüp küçültülmesi kötü görünmesine yol açacağı için **FixedToolWindow** olarak belirlenerek formun boyutlandırılması engellenir.

"StartPosition" özelliği, **CenterScreen** olarak belirlenir. Form, ekranın tam ortasında açılır.

TextBox Nesneleri

"Name" özellikleri; **txtNo, txtAd, txtSoyad, txtTelefon** ve son olarak arama bölümünde **txtOgrenciAra** şeklinde belirlenir.

ComboBox Nesneleri

"Name" özellikleri, **comboSinif** ve **comboCinsiyet** olarak belirlenir.

Sınıf bilgisi gösterilecek ComboBox nesnesine **9, 10, 11, 12** değerleri eklenir.

Cinsiyet bilgisi gösterilecek ComboBox nesnesine **Kız** ve **Erkek** değerleri eklenir.

Button Nesneleri

"Name" özellikleri; **btnKaydet, btnSil, btnGuncelle** olarak belirlenir.

"Text" ve "Image" özellikleri forma uygun olarak ayarlanır.

"Size" özelliği ile forma uygun olarak boyutlandırılır.

DataGridView Nesnesi

"Name" özelliği, **gridOgrenci** olarak belirlenir.

"EditMode" özelliği, **EditProgrammatically** olarak belirlenir.

"SelectionMode" özelliği, **FullRowSelect** olarak ayarlanır.

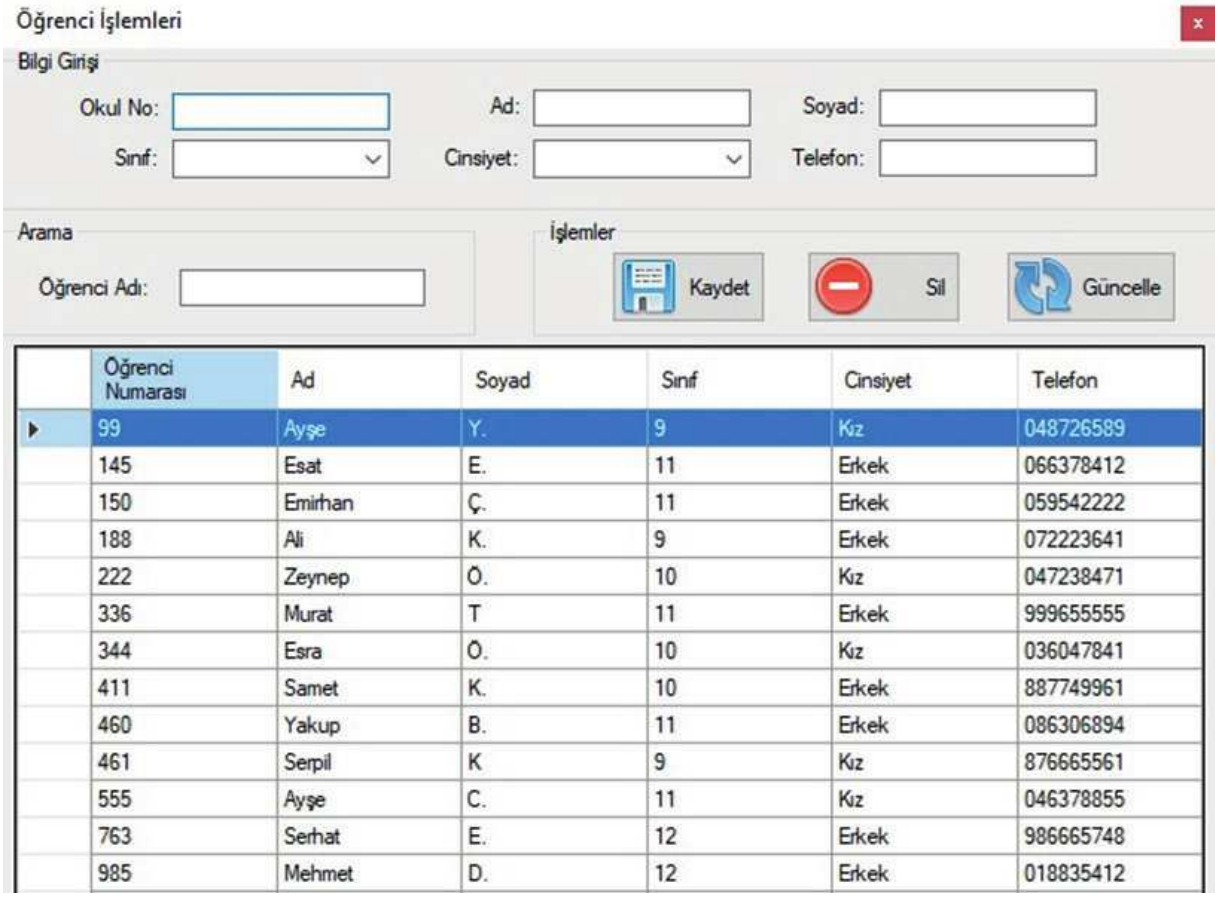
"AutoSizeColumnsMode" özelliği, **Fill** olarak ayarlanır.

Öğrenci Listeleme İşlemi

Öğrenci listeleme kodları bir metot içine yazılır ve bu metot, formun yüklenmesi olayında çağrılır (Görsel 6.103). Bu işlemden sonra öğrenci bilgilerinin DataGridView nesnesinde listelendiği görülür (Görsel 6.104).

```
VeriTabaniIslemleri vtIslemleri = new VeriTabaniIslemleri(); // VeriTabaniIslemleri sınıfından bir nesne örneği oluşturulur
MySqlConnection baglanti;
MySqlCommand komut;
string komutSatiri;
1 reference
private void formOgrenci_Load(object sender, EventArgs e)
{
    Listele(); // Alt bölümde yazılan listeleme metodu form yüklendiğinde çağırılır
}
4 references
public void Listele()
{
    try
    {
        baglanti = vtIslemleri.baglan(); // Veri tabanı bağlantı nesnesi oluşturulur
        komutSatiri = "Select * From ogrenciler"; // Verileri listeleyecek olan SQL sorgusu yazılır
        MySqlDataAdapter dataAdapter = new MySqlDataAdapter(komutSatiri,baglanti); // DataAdapter nesnesi oluşturulur.
        DataTable dataTable = new DataTable();
        dataAdapter.Fill(dataTable); // Sorgu sonucunda dönen kayıtlar DataTable nesnesine aktarılır
        gridOgrenci.DataSource = dataTable; // DataTable nesnesindeki kayıtlar DataGridView'de listelenir
        gridOgrenci.Columns["ogrenci_no"].HeaderText = "Öğrenci Numarası"; //DataGridView nesnesinde sütun başlıkları belirlenir
        gridOgrenci.Columns["ad"].HeaderText = "Ad";
        gridOgrenci.Columns["soyad"].HeaderText = "Soyad";
        gridOgrenci.Columns["sinif"].HeaderText = "Sınıf";
        gridOgrenci.Columns["cinsiyet"].HeaderText = "Cinsiyet";
        gridOgrenci.Columns["telefon"].HeaderText = "Telefon";
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Hata Oluştu", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Görsel 6.103: Öğrenci listeleme işlemi



Öğrenci Numarası	Ad	Soyad	Sınıf	Cinsiyet	Telefon
99	Ayşe	Y.	9	Kız	048726589
145	Esat	E.	11	Erkek	066378412
150	Emirhan	Ç.	11	Erkek	059542222
188	Ali	K.	9	Erkek	072223641
222	Zeynep	Ö.	10	Kız	047238471
336	Murat	T	11	Erkek	999655555
344	Esra	Ö.	10	Kız	036047841
411	Samet	K.	10	Erkek	887749961
460	Yakup	B.	11	Erkek	086306894
461	Serpil	K	9	Kız	876665561
555	Ayşe	C.	11	Kız	046378855
763	Serhat	E.	12	Erkek	986665748
985	Mehmet	D.	12	Erkek	018835412

Görsel 6.104: Verilerin DataGridView'de görüntülenmesi

Öğrenci Ekleme İşlemi

Öğrenci İşlemleri form ekranına veri girişi yapıldıktan sonra "Kaydet" butonuna tıklandığı zaman Görsel 6.105'teki kodlar çalıştırılır ve öğrenci kayıt işlemi gerçekleştirilir. Kodlar "Kaydet" butonunun tıklanma olayına yazılır. Öğrenci eklenirken aynı numaraya sahip bir öğrenci daha kaydedilmek istenirse program hata verecektir. Bunun nedeni, öğrenci_no alanının PK olarak belirlenmesidir. Kayıt işleminin başarılı olması için bilgilerin doğru bir şekilde girilmesi gerekir.

```
private void btnKaydet_Click(object sender, EventArgs e)
{
    try
    {
        if (baglanti.State != ConnectionState.Open) // Bağlantının durumu kontrol edilir
        {
            baglanti.Open(); // Eğer bağlantı açık değilse açılır
        }
        komutSatiri = "INSERT INTO ogrenciler (ogrenci_no,ad,soyad,sinif,cinsiyet,telefon) VALUES(@no,@ad,@soyad,@sinif,@cinsiyet,@telefon)";
        komut = new MySqlCommand(komutSatiri, baglanti); // Komut çalıştırmak için MySqlCommand nesnesi oluşturulur
        komut.Parameters.AddWithValue("@no", int.Parse(txtNo.Text)); // Sorguda verilen parametrelerin değerleri belirlenir
        komut.Parameters.AddWithValue("@ad", txtAd.Text);
        komut.Parameters.AddWithValue("@soyad", txtSoyad.Text);
        komut.Parameters.AddWithValue("@sinif", int.Parse(comboSinif.SelectedItem.ToString()));
        komut.Parameters.AddWithValue("@cinsiyet", comboCinsiyet.SelectedItem.ToString());
        komut.Parameters.AddWithValue("@telefon", txtTelefon.Text);

        komut.ExecuteNonQuery(); // Ekleme sorgusu çalıştırılır ve hata oluşmazsa öğrenci eklenir
        baglanti.Close(); // Bağlantı kapatılır
        Temizle(); // Form elemanlarının içerikleri temizlenir
        MessageBox.Show("İşlem başarılı", "Mesaj", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
        Listele(); // Eklenen öğrencinin DataGridView'de görülebilmesi için veriler tekrar listelenir
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Hata oluştu", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Görsel 6.105: Öğrenci ekleme işlemi

Form üzerinde ekleme, silme ve güncelleme işlemi yapıldıktan sonra form elemanlarının içindeki değerlerin silinmesi gerekir. Bunun için Temizle isimli bir metot oluşturulur ve gerekli yerlerde çağrılır (Görsel 6.106).

```
public void Temizle()
{
    txtAd.Clear();
    txtSoyad.Clear();
    txtNo.Clear();
    txtTelefon.Clear();
}
```

Görsel 6.106: Form içeriğinin temizlenmesi

Sıra Sizde

Öğrenci form ekranını kullanarak sınıfınızdaki öğrencileri sisteme ekleyiniz.

Öğrenci Silme İşlemi

Öğrenci silme ve güncelleme işlemleri yapılmadan önce öğrencinin DataGridView üzerinde seçilmesi ve o kayıt ile ilgili bilgilerin form elemanlarında gösterilmesi gerekir. Bunun için nesnenin "CellClick (Hücrenin tıklanması)" olayına Görsel 6.107'deki kodların yazılması gerekir.

```
private void gridOgrenci_CellClick(object sender, DataGridViewCellEventArgs e)
{
    try
    {
        // DataGridView'de seçili olan öğrenciye ait bilgiler form elemanlarına yazdırılır
        txtNo.Text = gridOgrenci.CurrentRow.Cells["ogrenci_no"].Value.ToString();
        txtAd.Text = gridOgrenci.CurrentRow.Cells["ad"].Value.ToString();
        txtSoyad.Text = gridOgrenci.CurrentRow.Cells["soyad"].Value.ToString();
        txtTelefon.Text = gridOgrenci.CurrentRow.Cells["telefon"].Value.ToString();
        comboSinif.SelectedItem = gridOgrenci.CurrentRow.Cells["sinif"].Value.ToString();
        comboCinsiyet.SelectedItem = gridOgrenci.CurrentRow.Cells["cinsiyet"].Value.ToString();
    }
    catch (Exception)
    {
        MessageBox.Show("Hata oluştu", "Mesaj", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Görsel 6.107: DataGridView nesnesindeki seçili kaydın form elemanlarına aktarılması

DataGridView üzerinde silinmesi istenen öğrenci seçildikten sonra o öğrenciye ait bilgiler form elemanlarında gösterilir. "Sil" butonuna tıklandığı zaman Görsel 6.108'deki kodlar çalıştırılır ve öğrenci bilgileri veri tabanından silinir. Kodlar "Sil" butonunun tıklanma olayına yazılır. "Ogrenciler" ve "odunc_kitaplar" tabloları arasında ilişki oluşturulduğu için ödünç kitap alan öğrenci sistemden silinemez.

```
private void btnSil_Click(object sender, EventArgs e)
{
    try
    {
        if (baglanti.State != ConnectionState.Open) // Bağlantı durumu kontrol edilir
        {
            baglanti.Open(); // Eğer açık değilse bağlantı açılır
        }
        komutSatiri = "DELETE FROM ogrenciler WHERE ogrenci_no = @no"; //@no yerine parametre gelecektir
        komut = new MySqlCommand(komutSatiri,baglanti);
        komut.Parameters.AddWithValue("@no", gridOgrenci.CurrentRow.Cells["ogrenci_no"].Value.ToString());
// Üst satırda sorguya parametre olarak DataGridView'de seçili olan ogrenci_no bilgisi gönderilir
        komut.ExecuteNonQuery(); // Sorgu çalıştırılır ve hata oluşmazsa öğrenci silinir
        baglanti.Close(); // Bağlantı kapatılır
        Temizle(); // Form elemanlarının içerikleri temizlenir.
        MessageBox.Show("İşlem başarılı", "Mesaj", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
        Listele(); // Silinen kayıtın görünmemesi için tekrar listeleme yapılır
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Hata oluştu", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

SIF

Görsel 6.108: Öğrenci silme işlemi

Öğrenci Güncelleme İşlemi

DataGridView üzerinde güncellenmesi istenen öğrenci seçildikten sonra o öğrenciye ait bilgiler form elemanlarında gösterilir. Yeni değerler girildikten sonra "Güncelle" butonuna tıklandığı zaman Görsel 6.109'daki kodlar çalıştırılır ve öğrenci bilgileri güncellenir. Kodlar "Güncelle" butonunun tıklanma olayına yazılır.

```
private void btnGuncelle_Click(object sender, EventArgs e)
{
    try
    {
        if (baglanti.State != ConnectionState.Open)
        {
            baglanti.Open();
        }
     komutSatiri = "UPDATE ogrenciler SET ad=@ad,soyad=@soyad,sinif=@sinif,cinsiyet=@cinsiyet,telefon=@telefon where ogrenci_no=@no";
        komut = new MySqlCommand(komutSatiri,baglanti);
        komut.Parameters.AddWithValue("@no", int.Parse(gridOgrenci.CurrentRow.Cells["ogrenci_no"].Value.ToString()));
        komut.Parameters.AddWithValue("@ad", txtAd.Text);
        komut.Parameters.AddWithValue("@soyad", txtSoyad.Text);
        komut.Parameters.AddWithValue("@sinif", int.Parse(comboSinif.SelectedItem.ToString()));
        komut.Parameters.AddWithValue("@cinsiyet", comboCinsiyet.SelectedItem.ToString());
        komut.Parameters.AddWithValue("@telefon", txtTelefon.Text);
        komut.ExecuteNonQuery();
        baglanti.Close();
        Temizle();
        MessageBox.Show("İşlem başarılı", "Mesaj", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
        Listele();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Hata oluştu", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Görsel 6.109: Öğrenci güncelleme işlemi

Sıra Sizde

Sistemde kayıtlı üç öğrencinin bilgilerini güncelleyiniz.

Öğrenci Arama İşlemi

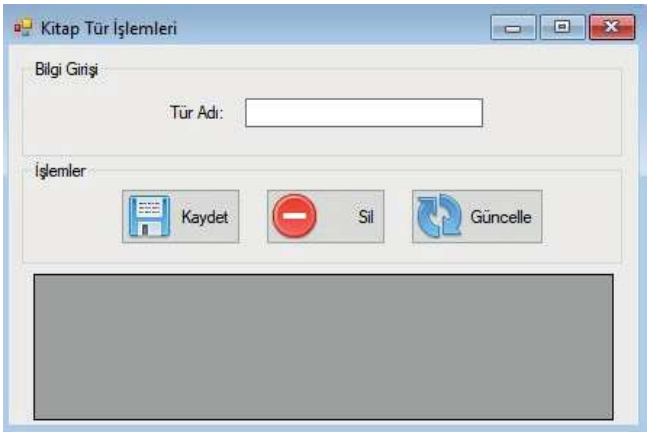
Form üzerindeki arama bölümünde öğrenci adına göre arama yapılabilmesini sağlayan kodlar Görsel 6.110'da görülür. Kodlar "TextChanged" olayına yazıldığı için arama kutusundaki her değişiklikte arama işlemi tekrarlanır ve sonuçlar DataGridView nesnesinde gösterilir.

```
private void txtAramaOgrenci_TextChanged(object sender, EventArgs e)
{
    OgrenciArama(txtAramaOgrenci.Text); // Arama metodu TextBox nesnesindeki her değişiklikte çağırılır
}
1 reference
public void OgrenciArama(string aranacakKelime)
{
    try
    {
        if (baglanti.State != ConnectionState.Open)
        {
            baglanti.Open();
        }
        komut = new MySqlCommand();// MySqlCommand nesnesinin parametreleri nesne oluşturulduktan sonra da belirtilebilir
        komut.Connection = baglanti; // MySqlCommand nesnesinin bağlantısı belirlenir
        komut.CommandText = "Select * From ogrenciler Where ad LIKE '"+aranacakKelime+"%'"; // Çalıştırılacak arama sorgusu
        MySqlDataAdapter dataAdapter = new MySqlDataAdapter(komut);
        DataTable dataTable = new DataTable();
        dataAdapter.Fill(dataTable); // Sorgudan dönen değer DataTable nesnesine doldurulur
        baglanti.Close();
        gridOgrenci.DataSource = dataTable; // DataTable içerisine aktarılan veriler DataGridView'de listelenir.

    }
    catch (Exception ex)
    {

        MessageBox.Show(ex.Message, "Hata oluştu", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }

}
```

Görsel 6.110: Öğrenci arama işlemi

6.8.5. Kitap Tür İşlemleri Sayfasının Hazırlanması

Kitap Tür İşlemleri sayfası; kitap türleri ile ilgili ekleme, silme, güncelleme, listeleme gibi işlemlerin yapılacağı sayfadır. Bu sayfada bulunan türler, kitap ekleme bölümünde kitaba ait tür olarak seçilir. Yeni bir form oluşturularak formKitapTur adı verilir. Ardından Görsel 6.111'deki form tasarımı yapılır. Formun alt bölümdeki "DataGridView" nesnesi, verilerin listelenmesi için kullanılır.



Görsel 6.111: Kitap Türleri form ekranı

Form nesnesinde ve içindeki form elemanlarında bazı özelliklerin ayarlanması gerekir.

Form Nesnesi

"Text" özelliği, **Kitap Tür İşlemleri** olarak belirlenir.

"Name" özelliği, **formKitapTur** olarak belirlenir.

"Size" özelliği, **450;500** olarak belirlenir.

"FormBorderStyle" özelliği, form ekranının büyütülüp küçültülmesi kötü görünmesine yol açacağı için **FixedToolWindow** olarak belirlenerek formun boyutlandırılması engellenir.

"StartPosition" özelliği, **CenterScreen** olarak belirlenir. Form, ekranın tam ortasında açılır.

TextBox Nesnesi

"Name" özelliği, **txtTurAdi** olarak belirlenir.

Button Nesneleri

"Name" özellikleri; **btnKaydet, btnSil, btnGuncelle** olarak belirlenir.

"Text" ve "Image" özellikleri forma uygun olarak ayarlanır.

"Size" özelliği ile forma uygun olarak boyutlandırılır.

DataGridView Nesnesi

"Name" özelliği, **gridKitapTur** olarak belirlenir.

"EditMode" özelliği, **EditProgrammatically** olarak belirlenir.

"SelectionMode" özelliği, **FullRowSelect** olarak ayarlanır.

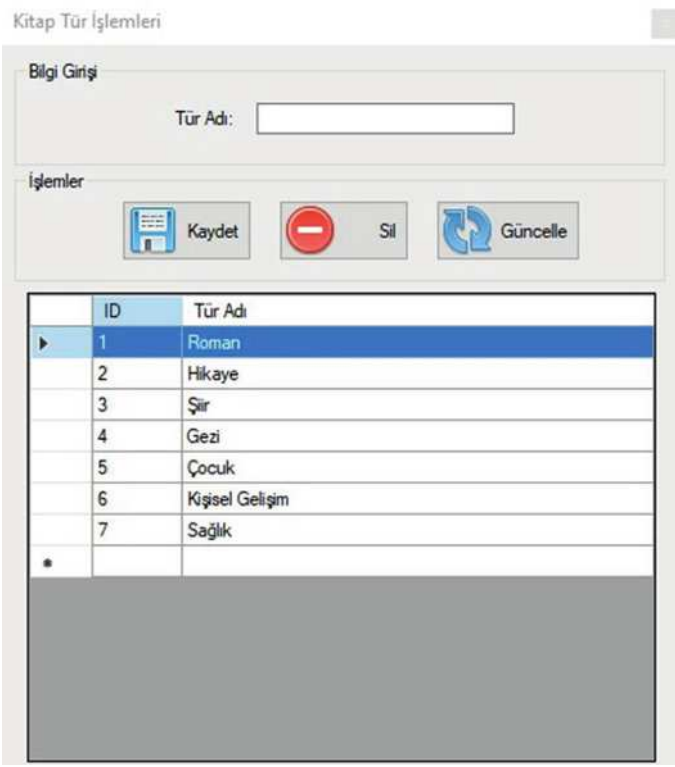
"AutoSizeColumnsMode" özelliği, **Fill** olarak ayarlanır.

Kitap Türlerini Listeleme

Kitap türlerini listeleme kodları bir metot içine yazılır ve bu metot, formun yüklenmesi olayında çağrılır (Görsel 6.112). Bu işlemden sonra kitap türlerinin DataGridView nesnesinde listelendiği görülür (Görsel 6.113).

```
VeriTabaniIslemleri vtIslemleri = new VeriTabaniIslemleri();
MySqlConnection baglanti;
MySqlCommand komut;
string komutSatiri;
1 reference
private void formKitapTur_Load(object sender, EventArgs e)
{
    TurleriListele();
}
4 references
public void TurleriListele()
{
    try
    {
        baglanti = vtIslemleri.baglan();
        komutSatiri = "Select * From kitap_turleri";
        MySqlDataAdapter dataAdapter = new MySqlDataAdapter(komutSatiri,baglanti);
        DataTable dataTable = new DataTable();
        dataAdapter.Fill(dataTable);
        gridKitapTur.DataSource = dataTable;
        gridKitapTur.Columns["tur_id"].HeaderText = "ID";
        gridKitapTur.Columns["tur_id"].Width = 100; // Sütun genişliği belirlenir
        gridKitapTur.Columns["tur_adi"].HeaderText = "Tür Adı";
        gridKitapTur.Columns["tur_adi"].Width = 300; // Sütun genişliği belirlenir
    }
    catch (Exception ex)
    {   MessageBox.Show(ex.Message, "Hata Oluştu", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Görsel 6.112: Kitap türleri listeleme işlemi



Görsel 6.113: Verilerin DataGridView'de görüntülenmesi

Kitap Türünü Ekleme İşlemi

Kitap Tür İşlemleri form ekranına tür adı bilgisi girişi yapıldıktan sonra "Kaydet" butonuna tıklandığında Görsel 6.114'teki kodlar çalıştırılır ve kitap türü kayıt işlemi gerçekleştirilir. Ekleme kodları "Kaydet" butonunun tıklanma olayına yazılır.

```
private void btnKaydet_Click(object sender, EventArgs e)
{
    try
    {
        if (baglanti.State != ConnectionState.Open)
        {
            baglanti.Open();
        }
        komut = new MySqlCommand();
        komut.Connection = baglanti;
        komut.CommandText = "INSERT INTO kitap_turleri (tur_adi) VALUES(@tur_adi)";
        komut.Parameters.AddWithValue("@tur_adi", txtTurAdi.Text);

        komut.ExecuteNonQuery();
        baglanti.Close();
        txtTurAdi.Clear();
        MessageBox.Show("İşlem başarılı", "Mesaj", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

        TurleriListele();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Hata Oluştu", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }

}
```

Görsel 6.114: Kitap türü ekleme işlemi

Sıra Sizde

Kitap Tür İşlemleri form ekranını kullanarak üç adet kitap türünü sisteme ekleyiniz.

Kitap Türünü Silme İşlemi

Kitap türünü silme ve güncelleme işlemleri yapılmadan önce kitap türünün DataGridView üzerinde seçilmesi ve o kayıt ile ilgili bilginin TextBox nesnesinde gösterilmesi gerekir. Bunun için nesnenin "CellClick" olayına Görsel 6.115'teki kodlar yazılır.

```
private void gridListe_CellClick(object sender, DataGridViewCellEventArgs e)
{
    try
    {
        txtTurAdi.Text = gridKitapTur.CurrentRow.Cells["tur_adi"].Value.ToString();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Hata Oluştu", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Görsel 6.115: DataGridView nesnesindeki seçili türün TextBox'a aktarılması

DataGridView üzerinde silinmesi istenen kitap türü seçildikten sonra o türün adı TextBox nesnesinde gösterilir. "Sil" butonuna tıklandığı zaman Görsel 6.116'daki kodlar çalıştırılır ve kayıt, veri tabanından silinir. Kodlar "Sil" butonunun tıklanma olayına yazılır. "Kitaplar" ve "kitap_turleri" tabloları arasında ilişki oluşturulduğu için kayıtlı bir kitaba ait tür sistemden silinemez.

```
private void btnSil_Click(object sender, EventArgs e)
{
    try
    {
        if (baglanti.State != ConnectionState.Open)
        {
            baglanti.Open();
        }
        komut = new MySqlCommand();
        komut.Connection = baglanti;
        komut.CommandText = "DELETE FROM kitap_turleri WHERE tur_id = @tur_id";

        komut.Parameters.AddWithValue("@tur_id", gridKitapTur.CurrentRow.Cells["tur_id"].Value.ToString());
        komut.ExecuteNonQuery();
        baglanti.Close();
        txtTurAdi.Clear();
        MessageBox.Show("İşlem başarılı", "Mesaj", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
        TurleriListele();
    }
    catch (Exception ex)
    {

        MessageBox.Show(ex.Message, "Hata Oluştu", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Görsel 6.116: Kitap türü silme işlemi

Sıra Sizde

Sisteme iki farklı kitap türü ekleyiniz ve daha sonra bu türleri sistemden siliniz.

Kitap Türünü Güncelleme İşlemi

DataGridView üzerinde güncellenmesi istenen kitap türü seçildikten sonra o türün adı TextBox nesnesinde gösterilir. Yeni değer girildikten sonra "Güncelle" butonuna tıklandığı zaman Görsel 6.117'deki kodlar çalıştırılır ve kitap türünü güncelleme işlemi gerçekleştirilir. Kodlar "Güncelle" butonunun tıklanma olayına yazılır.

```
private void btnGuncelle_Click(object sender, EventArgs e)
{
    try
    {
        if (baglanti.State != ConnectionState.Open)
        {
            baglanti.Open();
        }
        komut = new MySqlCommand();
        komut.Connection = baglanti;
        komut.CommandText = "UPDATE kitap_turleri SET tur_adi=@tur_adi where tur_id=@tur_id";
        komut.Parameters.AddWithValue("@tur_id", int.Parse(gridKitapTur.CurrentRow.Cells["tur_id"].Value.ToString()));
        komut.Parameters.AddWithValue("@tur_adi", txtTurAdi.Text);

        komut.ExecuteNonQuery();
        baglanti.Close();
        txtTurAdi.Clear();
        MessageBox.Show("İşlem başarılı", "Mesaj", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

        TurleriListele();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Hata Oluştu", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Görsel 6.117: Kitap türü güncelleme işlemi

Sıra Sizde

Sistemde kayıtlı olan iki adet kitap türünü güncelleyiniz.

6.8.6. Kitap İşlemleri Sayfasının Hazırlanması

Kitap İşlemleri sayfası; kütüphanedeki kitaplar ile ilgili ekleme, silme, güncelleme, listeleme ve arama gibi işlemlerin yapılacağı sayfadır. Yeni bir form oluşturularak formKitap adı verilir. Ardından Görsel 6.118'deki form tasarımı yapılır. Formun alt bölümündeki "DataGridView" nesnesi, verilerin listelenmesi için kullanılır.

Görsel 6.118: Kitap İşlemleri form tasarımı

Form nesnesinde ve içindeki form elemanlarında bazı özelliklerin ayarlanması gerekir.

Form Nesnesi

"Text" özelliği, **Kitap İşlemleri** olarak belirlenir.

"Name" özelliği, **formKitap** olarak belirlenir.

"Size" özelliği, **670;550** olarak belirlenir.

"FormBorderStyle" özelliği, form ekranının büyütülüp küçültülmesi kötü görünmesine yol açacağı için **FixedToolWindow** olarak belirlenerek formun boyutlandırılması engellenir.

"StartPosition" özelliği, **CenterScreen** olarak belirlenir. Form, ekranın tam ortasında açılır.

TextBox Nesneleri

"Name" özellikleri; **txtKitapAdi, txtYazar, txtYayinEvi, txtSayfaSayisi** ve son olarak arama bölümünde **txtKitapAra** şeklinde belirlenir.

ComboBox Nesnesi

"Name" özelliği, **comboKitapTur** olarak belirlenir.

ComboBox nesnesinin içine kitap_turleri tablosundaki veriler listelenir. DisplayMember (Gösterilecek alan) tur_adi, ValueMember (Değer alanı) tur_id olarak belirlenir. Kullanıcı tarafından kitap türünün adı görüntülenir fakat veri tabanına tur_id değeri kaydedilir.

Button Nesneleri

"Name" özellikleri; **btnKaydet, btnSil, btnGuncelle** olarak belirlenir.

"Text" ve "Image" özellikleri forma uygun olarak ayarlanır.

"Size" özelliği ile forma uygun olarak boyutlandırılır.

DataGridView Nesnesi

"Name" özelliği, **gridKitap** olarak belirlenir.

"EditMode" özelliği, **EditProgrammatically** olarak belirlenir.

"SelectionMode" özelliği, **FullRowSelect** olarak ayarlanır.

"AutoSizeColumnsMode" özelliği, **Fill** olarak ayarlanır.

Kitap ve Kitap Türlerini (Combobox) Listeleme İşlemi

ComboBox nesnesinde kitap türlerinin listelenmesi için gerekli olan kodlar **KitapTurYukle** metodunun içine yazılır (Görsel 6.119).

```
VeriTabaniIslemleri vtIslemleri = new VeriTabaniIslemleri();
MySqlConnection baglanti;
MySqlCommand komut;
string komutSatiri;
1 reference
private void formKitap_Load(object sender, EventArgs e)
{
    KitapTurYukle();
    KitapListele();
}
1 reference
public void KitapTurYukle()
{
    try
    {
        baglanti = vtIslemleri.baglan();
        komutSatiri = "Select * From kitap_turleri";
        MySqlDataAdapter dataAdapter = new MySqlDataAdapter(komutSatiri,baglanti);
        DataTable dataTable = new DataTable();
        dataAdapter.Fill(dataTable);
        comboKitapTur.DataSource = dataTable; // ComboBox nesnesinin veri kaynağı ayarlanır
        comboKitapTur.ValueMember = "tur_id"; // Arka planda tutulup veri tabanına kaydedilecek alan belirlenir
        comboKitapTur.DisplayMember = "tur_adi"; // Kullanıcıya gösterilecek alan belirlenir
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Hata Oluştu", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

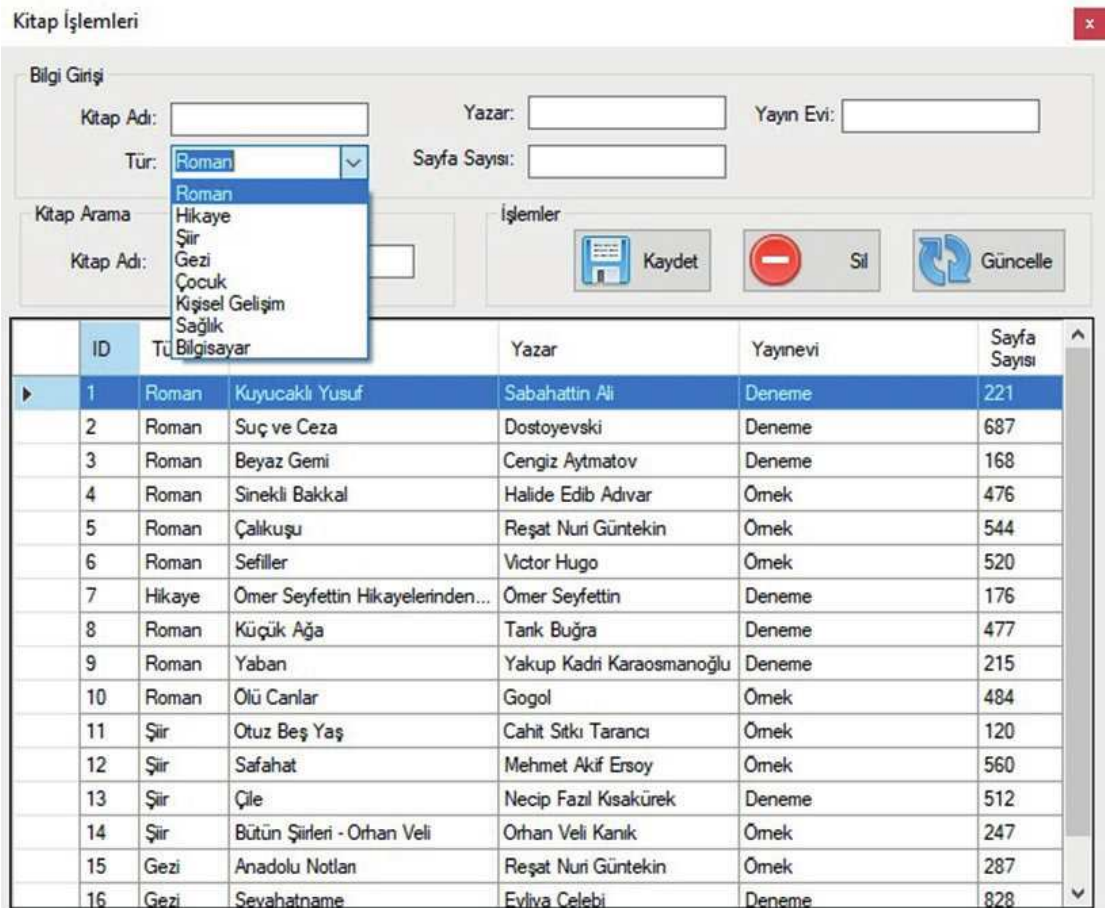
Görsel 6.119: Kitap türlerini ComboBox'ta listeleme işlemi

DataGridView nesnesinde kitapların listelenmesi için gerekli olan kodlar KitapListele metodunun içine yazılır (Görsel 6.120). Daha sonra bu iki metot, formun yüklenmesi olayında çağrılır.

```
public void KitapListele()
{
    try
    {
        baglanti = vtIslemleri.baglan();
komutSatiri = "Select kitap_id,tur_adi,kitap_adi,yazar,yayinevi,sayfa_sayisi From kitaplar,kitap_turleri where kitaplar.tur_id=kitap_turleri.tur_id";
        MySqlDataAdapter dataAdapter = new MySqlDataAdapter(komutSatiri, baglanti);
        DataTable dataTable = new DataTable();
        dataAdapter.Fill(dataTable);

        gridKitap.DataSource = dataTable;

        gridKitap.Columns["kitap_id"].HeaderText = "ID";
        gridKitap.Columns["kitap_id"].Width = 20;
        gridKitap.Columns["tur_adi"].HeaderText = "Tür";
        gridKitap.Columns["tur_adi"].Width = 30;
        gridKitap.Columns["kitap_adi"].HeaderText = "Adı";
        gridKitap.Columns["kitap_adi"].Width = 90;
        gridKitap.Columns["yazar"].HeaderText = "Yazar";
        gridKitap.Columns["yazar"].Width = 80;
        gridKitap.Columns["yayinevi"].HeaderText = "Yayınevi";
        gridKitap.Columns["yayinevi"].Width = 80;
        gridKitap.Columns["sayfa_sayisi"].HeaderText = "Sayfa Sayısı";
        gridKitap.Columns["sayfa_sayisi"].Width = 50;
    }
    catch (Exception ex)
    {

        MessageBox.Show(ex.Message, "Hata Oluştu", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }

}
```

Görsel 6.120: Kitap listeleme işlemi

Bu işlemlerden sonra kitap bilgilerinin DataGridView nesnesinde, kitap türlerinin de ComboBox nesnesinde listelendiği görülür (Görsel 6.121).



Görsel 6.121: Kitapların ve kitap türlerinin form ekranında listelenmesi

Kitap Ekleme İşlemi

Kitap İşlemleri form ekranına veri girişi yapıldıktan sonra "Kaydet" butonuna tıklandığı zaman Görsel 6.122'deki kodlar çalıştırılır ve kitap kayıt işlemi gerçekleştirilir. Kodlar "Kaydet" butonunun tıklanma olayına yazılır. Kayıt işleminin başarılı olması için bilgilerin doğru bir şekilde girilmesi gerekir.

```
private void btnKaydet_Click(object sender, EventArgs e)
{
    try
    {
        if (baglanti.State != ConnectionState.Open) // Bağlantının durumu kontrol edilir
        {
            baglanti.Open(); // Eğer bağlantı açık değilse açılır
        }
        komut = new MySqlCommand();
        komut.Connection = baglanti;
        komut.CommandText = "INSERT INTO kitaplar (tur_id,kitap_adi,yazar,yayinevi,sayfa_sayisi) " +
            "VALUES(@tur_id,@kitap_adi,@yazar,@yayinevi,@sayfa_sayisi)";
        // Alttaki satırda ComboBox nesnesinin arka planda tuttuğu id değeri tur_id olarak kayıt edilir
        komut.Parameters.AddWithValue("@tur_id", int.Parse(comboKitapTur.SelectedValue.ToString()));
        komut.Parameters.AddWithValue("@kitap_adi", txtKitapAdi.Text);
        komut.Parameters.AddWithValue("@yazar", txtYazar.Text);
        komut.Parameters.AddWithValue("@yayinevi", txtYayinEvi.Text);
        komut.Parameters.AddWithValue("@sayfa_sayisi", int.Parse(txtSayfaSayisi.Text));
        komut.ExecuteNonQuery();
        baglanti.Close();
        Temizle();
        MessageBox.Show("İşlem başarılı", "Mesaj", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
        KitapListele();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Hata Oluştu", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Görsel 6.122: Kitap ekleme işlemi

Kitap İşlemleri formu üzerinde ekleme, silme ve güncelleme işlemi yapıldıktan sonra form elemanlarının içindeki değerlerin silinmesi gerekir. Bunun için Temizle isimli bir metot oluşturulur ve gerekli yerlerde çağrılır (Görsel 6.123).

```
public void Temizle()
{
    txtKitapAdi.Clear();
    txtSayfaSayisi.Clear();
    txtYayinEvi.Clear();
    txtYazar.Clear();
}
```

Görsel 6.123: Form içeriğinin temizlenmesi

Sıra Sizde

Kitap form ekranını kullanarak farklı kategorilerde 10 adet kitabı sisteme ekleyiniz.

Kitap Silme İşlemi

Kitap silme ve güncelleme işlemleri yapılmadan önce işlem yapılacak kitabın DataGridView üzerinde seçilmesi ve o kayıt ile ilgili bilgilerin form elemanlarında gösterilmesi gerekir. Bunun için nesnenin "CellClick" olayına Görsel 6.124'teki kodların yazılması gerekir.

```
private void gridKitap_CellClick(object sender, DataGridViewCellEventArgs e)
{
    try
    {
        txtKitapAdi.Text = gridKitap.CurrentRow.Cells["kitap_adi"].Value.ToString();
        txtSayfaSayisi.Text = gridKitap.CurrentRow.Cells["sayfa_sayisi"].Value.ToString();
        txtYayinEvi.Text = gridKitap.CurrentRow.Cells["yayinevi"].Value.ToString();
        txtYazar.Text = gridKitap.CurrentRow.Cells["yazar"].Value.ToString();
        comboKitapTur.Text = gridKitap.CurrentRow.Cells["tur_adi"].Value.ToString();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Hata Oluştu", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Görsel 6.124: DataGridView nesnesindeki seçili kaydın form elemanlarına aktarılması

DataGridView üzerinde silinmesi istenen kitap seçildikten sonra o kitaba ait bilgiler form elemanlarında gösterilir. "Sil" butonuna tıklandığı zaman Görsel 6.125'teki kodlar çalıştırılır ve kitap bilgileri veri tabanından silinir. Kodlar "Sil" butonunun tıklanma olayına yazılır. Ödünç verilen bir kitap tablolar arası bağlantıdan dolayı silinemez.

```
private void btnSil_Click(object sender, EventArgs e)
{
    try
    {
        if (baglanti.State != ConnectionState.Open)
        {
            baglanti.Open();
        }
        komut = new MySqlCommand();
        komut.Connection = baglanti;
        komut.CommandText = "DELETE FROM kitaplar WHERE kitap_id = @kitap_id";

        komut.Parameters.AddWithValue("@kitap_id", gridKitap.CurrentRow.Cells["kitap_id"].Value.ToString());
        komut.ExecuteNonQuery();
        baglanti.Close();
        Temizle();
        MessageBox.Show("İşlem başarılı", "Mesaj", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
        KitapListele();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Hata Oluştu", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Görsel 6.125: Kitap silme işlemi

Sıra Sizde

Üç farklı öğrenciyi sisteme ekleyiniz ve daha sonra bu öğrencileri sistemden siliniz.

Kitap Bilgilerini Güncelleme İşlemi

DataGridView üzerinde güncellenmesi istenen kitap seçildikten sonra o kitaba ait bilgiler form elemanlarında gösterilir. Yeni değerler girildikten sonra "Güncelle" butonuna tıklandığı zaman Görsel 6.126'daki kodlar çalıştırılır ve kitap bilgileri güncellenir. Kodlar "Güncelle" butonunun tıklanma olayına yazılır.

```
private void btnGuncelle_Click(object sender, EventArgs e)
{
    try
    {
        if (baglanti.State != ConnectionState.Open)
        {
            baglanti.Open();
        }
        komut = new MySqlCommand();
        komut.Connection = baglanti;
        komut.CommandText = "UPDATE kitaplar SET tur_id=@tur_id,kitap_adi=@kitap_adi," +
            "yazar=@yazar,yayinevi=@yayinevi,sayfa_sayisi=@sayfa_sayisi where kitap_id=@kitap_id";
        komut.Parameters.AddWithValue("@kitap_id", int.Parse(gridKitap.CurrentRow.Cells["kitap_id"].Value.ToString()));
        komut.Parameters.AddWithValue("@tur_id", int.Parse(comboKitapTur.SelectedValue.ToString()));
        komut.Parameters.AddWithValue("@kitap_adi", txtKitapAdi.Text);
        komut.Parameters.AddWithValue("@yazar", txtYazar.Text);
        komut.Parameters.AddWithValue("@yayinevi", txtYayinEvi.Text);
        komut.Parameters.AddWithValue("@sayfa_sayisi", int.Parse(txtSayfaSayisi.Text));
        komut.ExecuteNonQuery();
        baglanti.Close();
        Temizle();
        MessageBox.Show("İşlem başarılı", "Mesaj", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
        KitapListele();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Hata Oluştu", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Görsel 6.126: Kitap güncelleme işlemi

Sıra Sizde

Sistemde kayıtlı olan üç adet kitabın bilgilerini güncelleyiniz.

Kitap Arama İşlemi

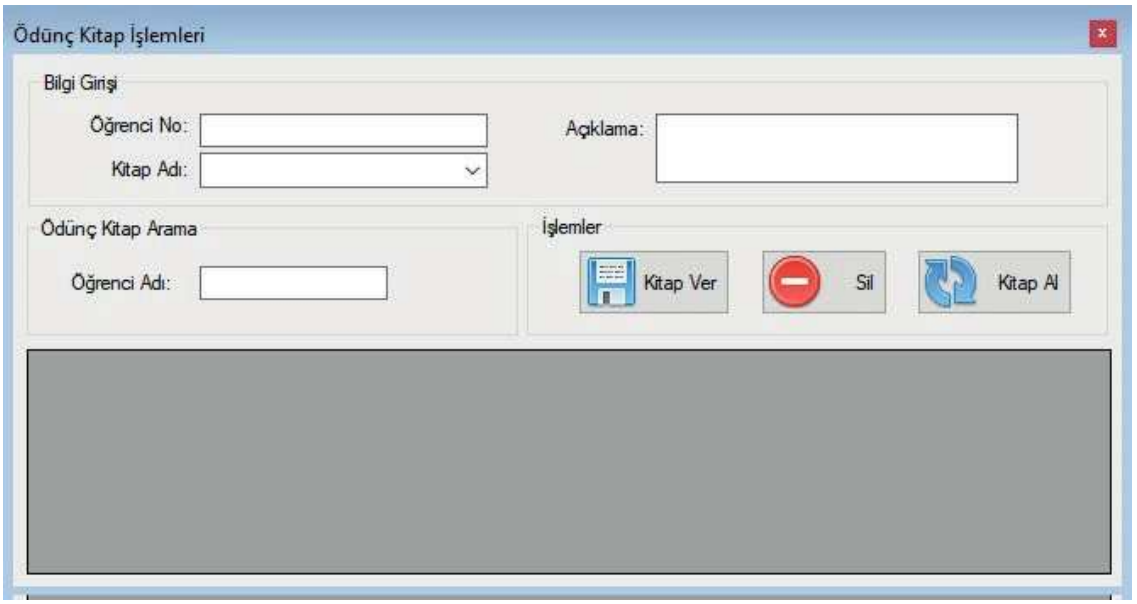
Form üzerindeki arama bölümünde kitap adına göre arama yapılabilmesini sağlayan kodlar Görsel 6.127'de görülür. Kodlar "TextChanged" olayına yazıldığı için arama kutusundaki her değişiklikte arama işlemi tekrarlanır ve sonuçlar DataGridView nesnesinde gösterilir.

```
private void txtKitapAra_TextChanged(object sender, EventArgs e)
{
    KitapArama(txtKitapAra.Text);
}
1 reference
public void KitapArama(string aranacakKelime)
{
    try
    {
        if (baglanti.State != ConnectionState.Open)
        {
            baglanti.Open();
        }
        komutSatiri = "Select kitap_id,tur_adi,kitap_adi,yazar,yayinevi,sayfa_sayisi From kitaplar,kitap_turleri " +
            "where kitaplar.tur_id=kitap_turleri.tur_id and kitap_adi LIKE '" + aranacakKelime + "%'";
        MySqlDataAdapter dataAdapter = new MySqlDataAdapter(komutSatiri,baglanti);
        DataTable dataTable = new DataTable();
        dataAdapter.Fill(dataTable);
        baglanti.Close();
        gridKitap.DataSource = dataTable;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Hata oluştu", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Görsel 6.127: Kitap arama işlemi

6.8.7. Ödünç Kitap İşlemleri Sayfasının Hazırlanması

Ödünç Kitap İşlemleri sayfası, kütüphanedeki kitapların öğrencilere ödünç olarak verilmesi ve geri alınması işlemlerinin yapılacağı sayfadır. Ödünç kitap verme ve geri alma işlemlerinde tarih olarak, sistem tarafından o günün tarihi otomatik olarak verilir. Yeni bir form oluşturularak formOduncKitap adı verilir. Ardından Görsel 6.128'deki form tasarımı yapılır. Formun alt bölümündeki "DataGridView" nesnesi, verilerin listelenmesi için kullanılır.



Görsel 6.128: Ödünç Kitap İşlemleri form tasarımı

Form nesnesinde ve içindeki form elemanlarında bazı özelliklerin ayarlanması gerekir.

Form Nesnesi

"Text" özelliği, **Ödünç Kitap İşlemleri** olarak belirlenir.

"Name" özelliği, **formOduncKitap** olarak belirlenir.

"Size" özelliği, **800;550** olarak belirlenir.

"FormBorderStyle" özelliği, form ekranının büyütülüp küçültülmesi kötü görünmesine yol açacağı için **FixedToolWindow** olarak belirlenerek formun boyutlandırılması engellenir.

"StartPosition" özelliği, **CenterScreen** olarak belirlenir. Form, ekranın tam ortasında açılır.

TextBox Nesneleri

"Name" özellikleri; **txtNo, txtAciklama** ve son olarak arama bölümünde **txtAramaOgrenci** şeklinde belirlenir.

ComboBox Nesnesi

"Name" özelliği, **comboKitap** olarak belirlenir.

ComboBox nesnesinin içine kitaplar tablosundaki veriler listelenir. DisplayMember (Gösterilecek alan) kitap_adi, ValueMember (Değer alanı) kitap_id olarak belirlenir. Kullanıcı tarafından kitap adları görüntülenir fakat veri tabanına kitap_id değeri kaydedilir.

Button Nesneleri

"Name" özellikleri; **btnKitapVer, btnSil, btnKitapAl** olarak belirlenir.

"Text" ve "Image" özellikleri forma uygun olarak ayarlanır.

"Size" özelliği ile forma uygun olarak boyutlandırılır.

DataGridView Nesnesi

"Name" özelliği, **gridOduncKitaplar** olarak belirlenir.

"EditMode" özelliği, **EditProgrammatically** olarak belirlenir.

"SelectionMode" özelliği, **FullRowSelect** olarak ayarlanır.

"AutoSizeColumnsMode" özelliği, **Fill** olarak ayarlanır.

Kitapları (ComboBox) ve Ödünç Alınan Kitapları Listeleme İşlemi

ComboBox nesnesinde kitapların listelenmesi için gerekli olan kodlar **KitapYukle** metodunun içine yazılır (Görsel 6.129).

```
VeriTabaniIslemleri vtIslemleri = new VeriTabaniIslemleri();
MySqlConnection baglanti;
MySqlCommand komut;
string komutSatiri;
1 reference
private void formOduncKitap_Load(object sender, EventArgs e)
{
    Listele();
    KitapYukle();
}
4 references
public void KitapYukle()
{
    try
    {
        komutSatiri = "select * from kitaplar where kitap_id not in (select kitap_id from odunc_kitaplar where teslim_tarihi IS NULL)";
        // comboKitap nesnesinde veriler, ödünç verilip geri getirilmemiş kitaplar yer almayacak şekilde listelenir
        MySqlDataAdapter dataAdapter = new MySqlDataAdapter(komutSatiri,baglanti);
        DataTable dataTable = new DataTable();
        dataAdapter.Fill(dataTable);
        comboKitap.DataSource = dataTable;
        comboKitap.ValueMember = "kitap_id";
        comboKitap.DisplayMember = "kitap_adi";
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Hata oluştu", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

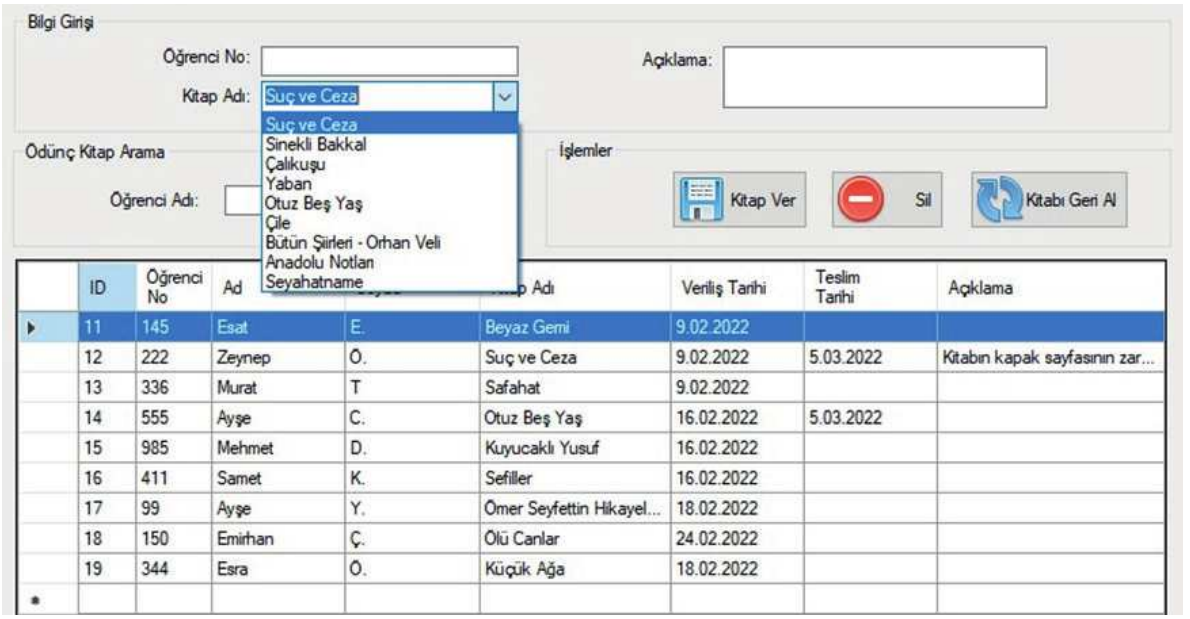
Görsel 6.129: Kitapları ComboBox'ta listeleme işlemi

DataGridView nesnesinde ödünç verilen kitapların listelenmesi için gerekli olan kodlar Listele metodunun içine yazılır (Görsel 6.130). Daha sonra bu iki metot, formun yüklenmesi olayında çağrılır.

```
public void Listele()
{
    try
    {
        baglanti = vtIslemleri.baglan();
        komutSatiri = "Select id,ogrenci_no,ad,soyad,kitap_adi,verilis_tarihi,teslim_tarihi,aciklama " +
            "From kitaplar,ogrenciler,odunc_kitaplar " +
            "where ogr_no=ogrenci_no and kitaplar.kitap_id=odunc_kitaplar.kitap_id";
        MySqlDataAdapter dataAdapter = new MySqlDataAdapter(komutSatiri, baglanti);
        DataTable dataTable = new DataTable();
        dataAdapter.Fill(dataTable);
        gridOduncKitaplar.DataSource = dataTable;

        gridOduncKitaplar.Columns["id"].HeaderText = "ID";
        gridOduncKitaplar.Columns["id"].Width = 30;
        gridOduncKitaplar.Columns["ogrenci_no"].HeaderText = "Öğrenci No";
        gridOduncKitaplar.Columns["ogrenci_no"].Width = 40;
        gridOduncKitaplar.Columns["ad"].HeaderText = "Ad";
        gridOduncKitaplar.Columns["ad"].Width = 70;
        gridOduncKitaplar.Columns["soyad"].HeaderText = "Soyad";
        gridOduncKitaplar.Columns["soyad"].Width = 70;
        gridOduncKitaplar.Columns["kitap_adi"].HeaderText = "Kitap Adı";
        gridOduncKitaplar.Columns["kitap_adi"].Width = 100;
        gridOduncKitaplar.Columns["verilis_tarihi"].HeaderText = "Veriliş Tarihi";
        gridOduncKitaplar.Columns["verilis_tarihi"].Width = 70;
        gridOduncKitaplar.Columns["teslim_tarihi"].HeaderText = "Teslim Tarihi";
        gridOduncKitaplar.Columns["teslim_tarihi"].Width = 70;
        gridOduncKitaplar.Columns["aciklama"].HeaderText = "Açıklama";
        gridOduncKitaplar.Columns["aciklama"].Width = 150;
    }
    catch (Exception ex)
    {

        MessageBox.Show(ex.Message, "Hata oluştu", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Görsel 6.130: Ödünç verilen kitapları listeleme işlemi

Bu işlemlerden sonra ödünç kitap verme işlemlerinin DataGridView nesnesinde, kitapların da ComboBox nesnesinde listelendiği görülür (Görsel 6.131).



Görsel 6.131: Kitapların ve ödünç verilen kitapların form ekranında listelenmesi

Ödünç Kitap Verme İşlemi

Ödünç Kitap İşlemleri form ekranına veri girişi yapıldıktan sonra "Kitap Ver" butonuna tıklandığı zaman Görsel 6.132'deki kodlar çalıştırılır ve öğrenciye ödünç kitap verme işlemi gerçekleştirilir. Kodlar "Kitap Ver" butonunun tıklanma olayına yazılır. Kayıt işleminin başarılı olması için bilgilerin doğru bir şekilde girilmesi gerekir.

```
private void btnKitapVer_Click(object sender, EventArgs e)
{
    try
    {
        if (baglanti.State != ConnectionState.Open) baglanti.Open();

        komut = new MySqlCommand();
        komut.Connection = baglanti;
        komut.CommandText = "INSERT INTO odunc_kitaplar (ogr_no,kitap_id,verilis_tarihi,aciklama) " +
            "VALUES(@ogr_no,@kitap_id,@verilis_tarihi,@aciklama)";
        komut.Parameters.AddWithValue("@ogr_no", int.Parse(txtNo.Text));
        komut.Parameters.AddWithValue("@kitap_id", int.Parse(comboKitap.SelectedValue.ToString()));
        komut.Parameters.AddWithValue("@verilis_tarihi", DateTime.Now.ToString("yyyy/MM/dd"));
        komut.Parameters.AddWithValue("@aciklama", txtAciklama.Text);
        komut.ExecuteNonQuery();
        baglanti.Close();
        temizle();
        KitapYukle();
        Listele();
        MessageBox.Show("İşlem başarılı", "Mesaj", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Hata oluştu", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Görsel 6.132: Ödünç kitap verme işlemi

Ödünç Kitap İşlemleri formu üzerinde işlem yapıldıktan sonra form elemanlarının içindeki değerlerin silinmesi gerekir. Bunun için temizle isimli bir metot oluşturulur ve gerekli yerlerde çağrılır (Görsel 6.133).

```
public void temizle()
{
    txtNo.Clear();
    txtAciklama.Clear();
}
```

Görsel 6.133: Form içeriğinin temizlenmesi

Sıra Sizde

Ödünç kitap işlemleri form ekranını kullanarak 10 tane ödünç kitap verme işlemi gerçekleştiriniz.

Ödünç Verilen Kitabı Silme İşlemi

Ödünç kitap silme işlemi yapılmadan önce işlem yapılacak kaydın DataGridView üzerinde seçilmesi gerekir. Bunun için nesnenin "CellClick" olayına Görsel 6.134'teki kodlar yazılmalıdır.

```
private void gridOduncKitaplar_CellClick(object sender, DataGridViewCellEventArgs e)
{
    try
    {
        txtAciklama.Text = gridOduncKitaplar.CurrentRow.Cells["aciklama"].Value.ToString();

    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Hata oluştu", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Görsel 6.134: DataGridView nesnesindeki seçili kaydın form elemanına aktarılması

DataGridView üzerinde silinmesi istenen kayıt seçildikten sonra o kayıt ile ilgili bilgiler form elemanlarında gösterilir. "Sil" butonuna tıklandığı zaman Görsel 6.135'teki kodlar çalıştırılır ve ödünç alma işlemi veri tabanından silinir. Kodlar "Sil" butonunun tıklanma olayına yazılır.

```
private void btnSil_Click(object sender, EventArgs e)
{
    try
    {
        if (baglanti.State != ConnectionState.Open) baglanti.Open();
        komut = new MySqlCommand();
        komut.Connection = baglanti;
        komut.CommandText = "DELETE FROM odunc_kitaplar WHERE id = @id";
        komut.Parameters.AddWithValue("@id", gridOduncKitaplar.CurrentRow.Cells["id"].Value.ToString());
        komut.ExecuteNonQuery();
        baglanti.Close();
        temizle();
        KitapYukle();
        Listele();

        MessageBox.Show("İşlem başarılı", "Mesaj", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Hata oluştu", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Görsel 6.135: Ödünç kitap verme işlemini silme

Sıra Sizde

İki adet ödünç kitap alma işlemini sistemden siliniz.

Ödünç Verilen Kitabı Geri Alma İşlemi

DataGridView üzerinde teslim alınacak ödünç kitap kaydı seçildikten sonra açıklama bölümü doldurularak "Kitabı Geri Al" butonu tıklandığında Görsel 6.136'daki kodlar çalıştırılır ve kitabı geri alma işlemi gerçekleştirilir.

```
private void btnKitapAl_Click(object sender, EventArgs e)
{
    try
    {
        if (gridOduncKitaplar.CurrentRow.Cells["teslim_tarihi"].Value.ToString()!=String.Empty)
        { // Teslim tarihi boş değilse yani kitap teslim alınmışsa işlem yapmadan Click olayından çıkılır
            MessageBox.Show("Bu kitap teslim alınmış", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return;
        }
        if (baglanti.State != ConnectionState.Open) baglanti.Open();
        komutSatiri = "UPDATE odunc_kitaplar SET teslim_tarihi=@teslim_tarihi,aciklama=@aciklama where id=@id";
        komut = new MySqlCommand(komutSatiri,baglanti);
        komut.Parameters.AddWithValue("@id", int.Parse(gridOduncKitaplar.CurrentRow.Cells["id"].Value.ToString()));
        komut.Parameters.AddWithValue("@teslim_tarihi", DateTime.Now.ToString("yyyy/MM/dd"));
        komut.Parameters.AddWithValue("@aciklama", txtAciklama.Text);
        komut.ExecuteNonQuery();
        baglanti.Close();
        temizle();
        KitapYukle();
        Listele();
        MessageBox.Show("İşlem başarılı", "Mesaj", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Hata oluştu", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Görsel 6.136: Ödünç verilen kitabı geri alma işlemi

Sıra Sizde

Ödünç olarak verilen beş kitabı teslim alma işlemini yapınız.

Ödünç Verilen Kitaplarda Arama İşlemi

Form üzerindeki arama bölümünde öğrenci adına göre arama yapılabilmesini sağlayan kodlar Görsel 6.137'de görülür. Kodlar "TextChanged" olayına yazıldığı için arama kutusundaki her değişiklikte arama işlemi tekrarlanır ve sonuçlar DataGridView nesnesinde gösterilir.

```
private void txtAramaOgrenci_TextChanged(object sender, EventArgs e)
{
    OduncKitapOgrenciArama(txtAramaOgrenci.Text);
}
1 reference
public void OduncKitapOgrenciArama(string aranacakKelime)
{
    try
    {
        if (baglanti.State != ConnectionState.Open) baglanti.Open();
        komutSatiri = "Select id,ogrenci_no,ad,soyad,kitap_adi,verilis_tarihi,teslim_tarihi,aciklama " +
            "From kitaplar,ogrenciler,odunc_kitaplar " +
            "where ogr_no=ogrenci_no and kitaplar.kitap_id=odunc_kitaplar.kitap_id and ad LIKE'" + aranacakKelime + "%'";
        MySqlDataAdapter dataAdapter = new MySqlDataAdapter(komutSatiri,baglanti);
        DataTable dataTable = new DataTable();
        dataAdapter.Fill(dataTable);
        gridOduncKitaplar.DataSource = dataTable;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Hata oluştu", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }

}
```

Görsel 6.137: Ödünç verilen kitaplarda arama işlemi

6.8.8. Kurulum (Setup) Hazırlanması

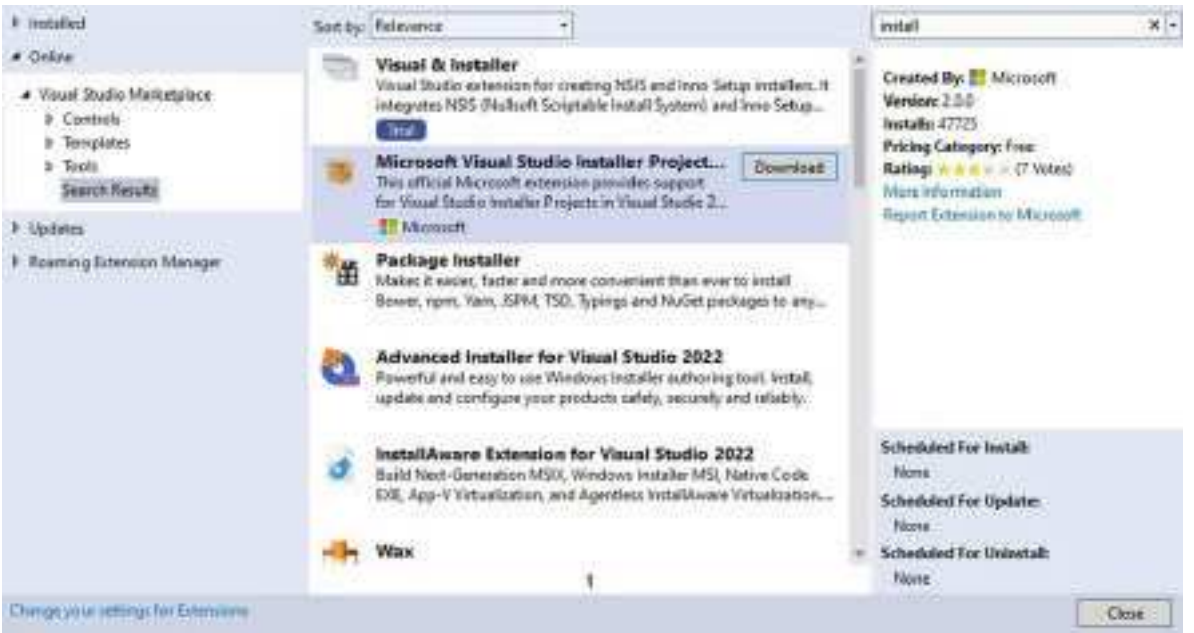
Geliştirilen projenin başka bilgisayarlarda kullanılabilmesi için Setup (Kurulum) hâline getirilmesi gerekir. Proje kurulum hâline getirildikten sonra farklı bir bilgisayara kurularak gerekli ayarlar yapıldıktan sonra kullanılabilir.

Not Programın hatasız çalışması için kurulum yapılacak bilgisayarda da MySQL Server kurulu olmalıdır. Projedeki veri tabanı, kurulum yapılacak bilgisayarın veri tabanına **import** edilmelidir.

Setup hazırlama adımları şunlardır:

1. Installer Projects Eklentisinin Kurulması

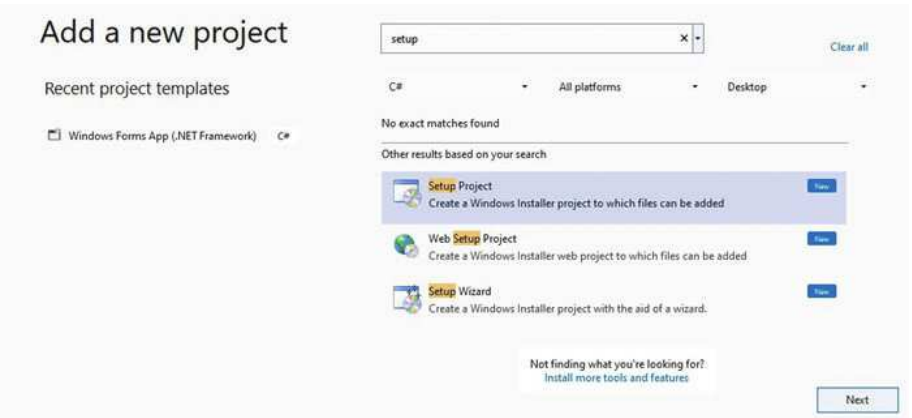
Extensions menüsünden "Manage Extensions" bölümüne tıklanır. Gelen pencerede arama bölümüne install yazılarak "Microsoft Visual Studio Installer Projects" eklentisinin görüntülenmesi sağlanır. Görüntülenen eklentideki **Download** düğmesine basılarak indirme işlemi başlatılır (Görsel 6.138). Kurulumun başlaması için kod editörü programı kapatılır.



Görsel 6.138: Visual Studio Installer Projects eklentisinin kurulması

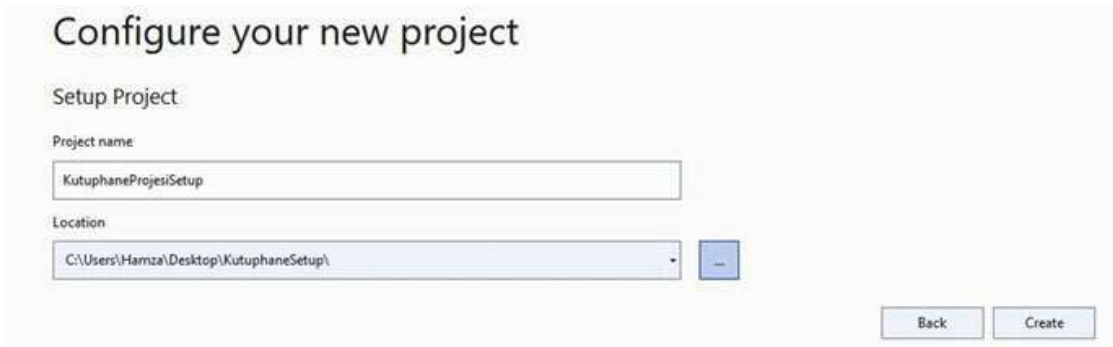
2. Setup Projesi Ekleme

Kurulum dosyası oluşturmak için setup projesinin eklenmesi gerekir. Bunun için **File** menüsünden Add-New Project seçeneği seçilir. Gelen pencereden "Setup Project" seçilerek diğer adıma geçilir (Görsel 6.139).



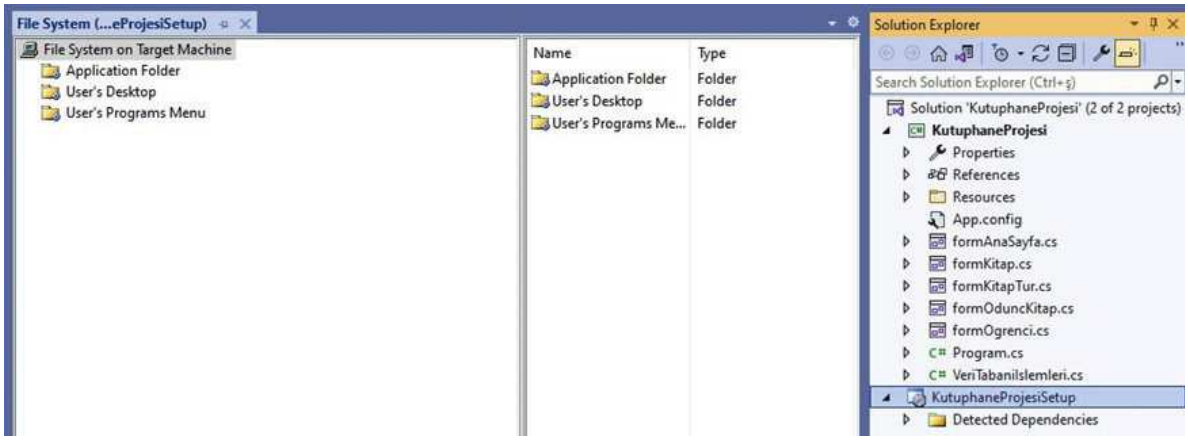
Görsel 6.139: Setup projesi oluşturma

Gelen pencereden Setup projesinin ismi belirlenir ve kurulum dosyalarının çıkarılacağı klasör seçilir (Görsel 6.140). Bu bölüm projeden bağımsız olarak kurulum ile ilgilidir. Örneğin Setup projesi, masaüstünde yeni bir klasör oluşturularak oraya çıkarılabilir. Create butonu tıklanarak Setup projesi oluşturulur.



Görsel 6.140: Setup projesi ayarları

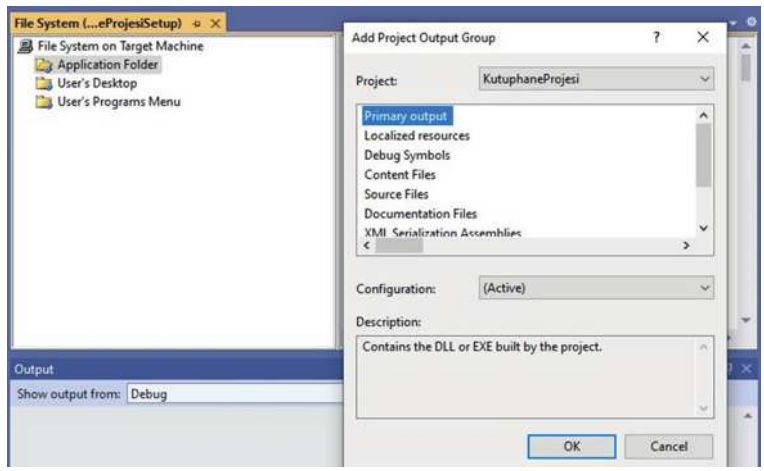
Bu işlemden sonra Setup projesi "Solution Explorer" penceresinde görüntülenir. Ayrıca Setup projesi ile ilgili işlemlerin yapılacağı bölüm de açılır (Görsel 6.141).



Görsel 6.141: Setup proje işlem penceresi

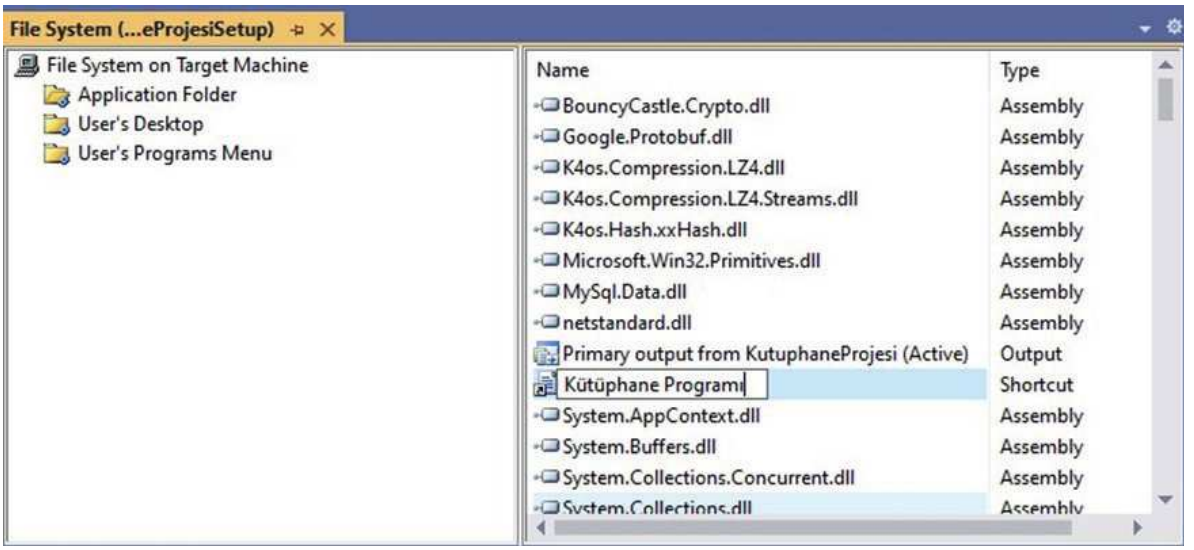
3. Proje Dosyalarının Setup Projesine Eklenmesi

Bu bölümde proje dosyalarının Setup projesine eklenmesi gerekir. Sağdaki bölümde "Application Folder" çift tıklanarak uygulama klasörüne giriş yapılır. Daha sonra sağ tıklanarak "Add - Project Output" seçeneği seçilir. Gelen ekranda "Primary output" seçilir (Görsel 6.142). Bu işlemden sonra projenin çalışması için gereken dosyaların tamamı Setup'a eklenir.



Görsel 6.142: Proje dosyalarının setup projesine eklenmesi

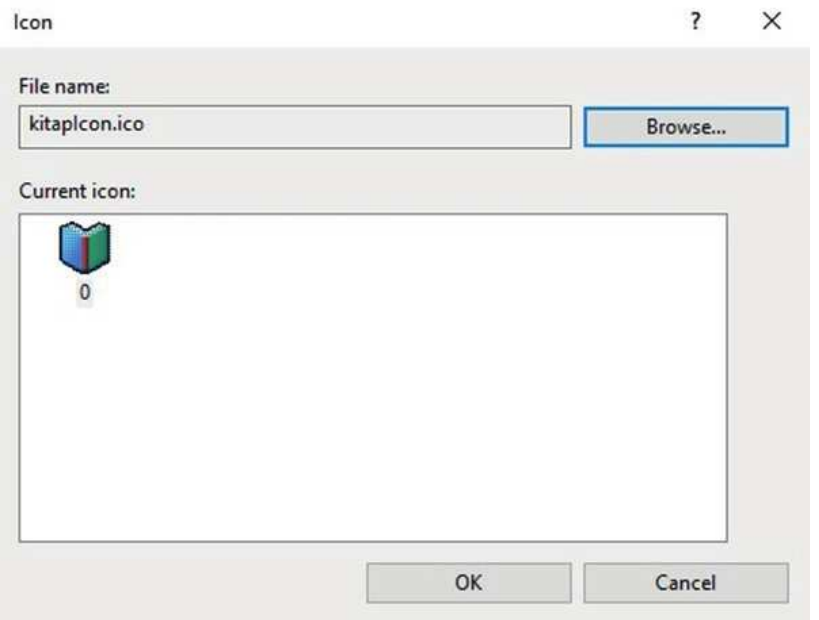
Setup'a eklenen ana dosya (output) olan **"Primary output from KutuphaneProjesi (Active)"** dosyasına sağ tıklanıp, "Create Shortcut to Primary output from KutuphaneProjesi (Active)" seçilerek kısayol oluşturulur. Bu işlem iki kez tekrarlanır. Oluşturulan kısayollara program adı (Kütüphane Programı) verilir çünkü bu kısayol dosyaları masaüstüne ve Başlat menüsüne eklenir. Bir başka deyişle proje, kurulduğu bilgisayarda bu isimle görüntülenir. Bu dosyalar sürükle bırak yöntemi ile "User's Desktop" ve "User's Programs Menu" klasörlerinin içine atılır (Görsel 6.143).



Görsel 6.143: Kısayol dosyalarının oluşturulması

4. Setup Projesine İkon Ekleme

Setup'a ikon eklenirken ikon dosyası da Setup içine alınmalıdır. Bunun için "Application Folder" içinde sağ tıklanarak "Add File" seçeneği ile ikon dosyası seçilir ve Setup projesine dâhil edilir. Daha sonra "User's Desktop" ve "User's Programs Menu" klasörlerinin içine atılan Kütüphane Programı kısayol dosyalarının özellikler bölümünden "Application Folder" içindeki ikon seçilir (Görsel 6.144).



Görsel 6.144: Setup projesine ikon eklenmesi

5. Setup Projesi Özelliklerini Belirleme

Setup dosyasının ayarları için Solution Explorer penceresinden Setup projesi seçilir ve Properties penceresine geçilir (Görsel 6.145).

KutuphaneProjesiSetup Deployment Project Properties

AddRemoveProgramsIcon	(Icon)
Author	Öğretmen
BackwardCompatibleIDGeneration	False
Description	Bu program ile kütüphanenizdeki ödünç kitap verme işlem
DetectNewerInstalledVersion	True
InstallAllUsers	True
Keywords	kütüphane programı
Localization	Turkish
Manufacturer	Milli Eğitim Bakanlığı
ManufacturerUrl	
PostBuildEvent	
PreBuildEvent	
ProductCode	{89CC737D-54AD-4A99-BCD3-87648E4E5DC6}
ProductName	KutuphaneProjesi
RemovePreviousVersions	True
RunPostBuildEvent	On successful build
SearchPath	
Subject	
SupportPhone	
SupportUrl	
TargetPlatform	x86
Title	Kütüphane Programı Kurulumu
UpgradeCode	{721CCD25-52D7-4A6C-AA77-B1FA6A351E64}
Version	1.0.0

Title
Specifies the title of an installer

Görsel 6.145: Setup projesi özellikleri

Setup projesinin özellikleri şunlardır:

AddRemoveProgramsIcon: Program ekle / kaldır kısmında görünen simge

Author: Programın kodlayıcısı

DetectNewerInstalledVersion: Programın daha yeni sürümü yüklü mü kontrol et.

InstallAllUsers: Program bütün kullanıcılar için kurulsun mu?

Keywords: Programla ilgili anahtar kelimeler

Localization: Bölgesel ayarlar (Localization, program Türkiye'de kullanılacağı için Türkiye seçilir.)

Manufacturer: Projeyi yapan firma adı

ManufacturerUrl: Projeyi yapan firmanın web adresi

ProductName: Projenin adı

RemovePreviousVersions: Eski sürümü varsa kaldır.

Subject: Programın kısa özeti

SupportPhone: Program desteği için telefon

SupportUrl: Program desteği için web adresi

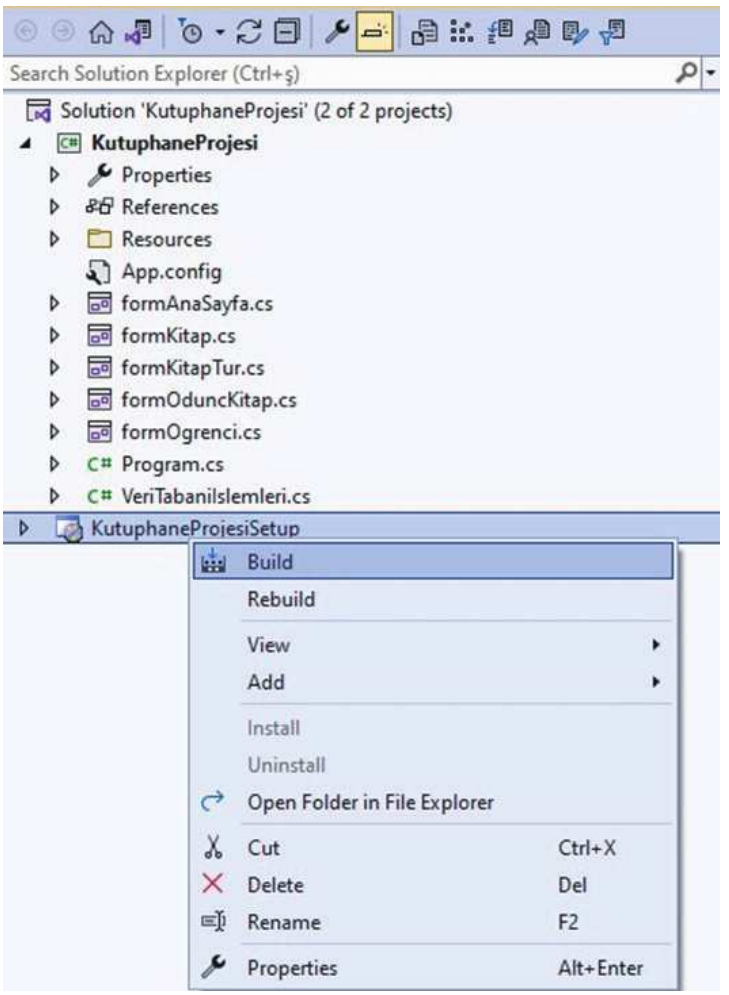
TargetPlatform: Programın desteklediği işlemci mimarisi

Title: Programın başlığı

Version: Programın versiyonu

6. Setup Projesinin Derlenmesi

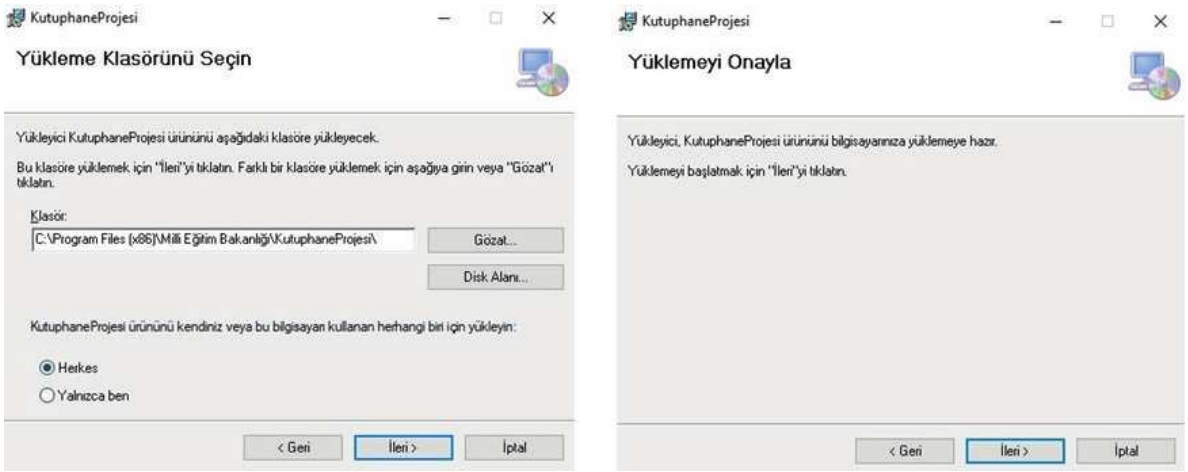
Artık Setup projesi derlenebilir. Görsel 6.146'da görüldüğü gibi Setup projesine sağ tıklanır ve "Build" seçeneği seçilir. Kurulum dosyalarının kaydedilmesi için seçilen klasörde iki adet Setup dosyası oluşur.



Görsel 6.146: Setup projesinin derlenmesi

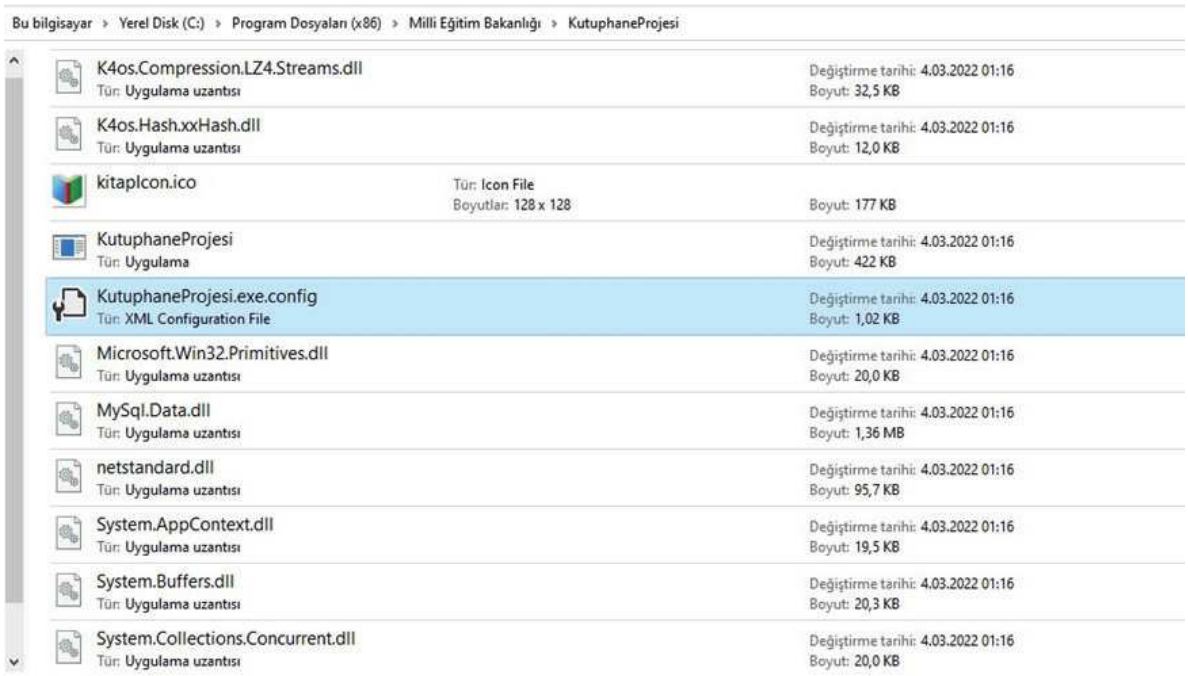
7. Program Kurulumunun Yapılması

Derleme işleminden sonra Setup projesinin debug klasörü içinde "KutuphaneProjesiSetup" ve "Setup" adında iki adet kurulum dosyasının oluştuğu görülür. Dosyalardan herhangi biri çalıştırılarak kurulum işlemi yapılır (Görsel 6.147). Uygulamanın çalışması için gerekli olan bir bileşen (Örneğin .NET Framework) kurulum yapılacak bilgisayarda yoksa program kurulumu uyarı verir ve eksik bileşenin kurulması sağlanır.



Görsel 6.147: Programın kurulum adımları

Program kurulumu yapıldıktan sonra masaüstü ekranında ve Başlat menüsünde "Kütüphane Programı" adı ve belirtilen ikonla birlikte uygulama dosyası görülür. Bu dosyalardan biri tıklanarak program çalıştırılır. Ayrıca programın kurulu olduğu klasörde program dosyaları da görüntülenir (Görsel 6.148).



Görsel 6.148: Kurulan programın dosyaları

Not Kurulum, projenin hazırlandığı bilgisayarda yapılırsa program sorunsuz bir şekilde çalışır fakat başka bir bilgisayara kurulduğunda veri tabanının MySQL sunucusuna dâhil edilmesi (import) ve **KutuphaneProjesi.exe.config** dosyasının içinde yer alan "Connection String"in kurulum yapılan bilgisayarın Server bilgilerine göre (sunucu adı, kullanıcı adı, şifre) güncellenmesi gerekir (Görsel 6.149).

*KutuphaneProjesi.exe - Not Defteri
Dosya Düzen Biçim Görünüm Yardım

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.8" />
  </startup>
  <connectionStrings>
    <add name="kutuphaneBaglantiCumlesi" connectionString="Server=localhost;Database=kutuphane;Uid=root;Pwd=19675561;" />
  </connectionStrings>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="System.Runtime.CompilerServices.Unsafe" publicKeyToken="b03f5f7f11d50a3a" culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-4.0.6.0" newVersion="4.0.6.0" />
      </dependentAssembly>
    </assemblyBinding>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="System.Buffers" publicKeyToken="cc7b13ffcd2ddd51" culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-4.0.3.0" newVersion="4.0.3.0" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```

Görsel 6.149: KutuphaneProjesi.exe.config dosyası

Not Proje tamamlandıktan sonra projenin bütün fonksiyonlarının çalışıp çalışmadığı kontrol edilir. Eksik veya yanlış yerler varsa düzeltilir ve tekrar proje setup dosyası hazırlanır.

Sıra Sizde

Projede hatalı veya eksik gördüğünüz yerleri tartışınız. Sonuçları aşağı yazınız.

1. .....
.....

2. .....
.....

3. .....
.....

4. .....
.....

Programda eksik gördüğünüz yerler varsa gerekli düzeltmeleri yaparak projeyi tekrar setup hâline getiriniz.

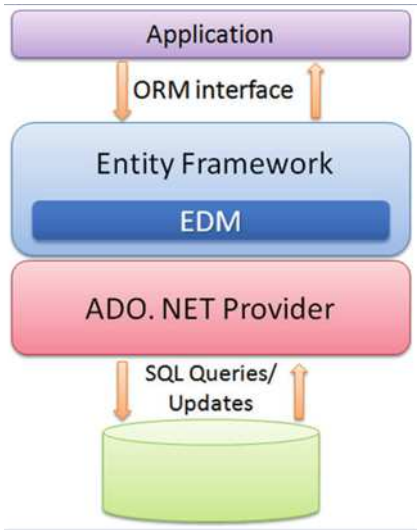
Sıra Sizde

Windows form ile ürün satışı ve stok takibi yapan bir form uygulaması yapınız.

- Ürünler, müşteriler, siparişlerin kaydedilmesi için gerekli tabloları ve alt tabloları içeren ilişkisel bir veri tabanı oluşturunuz.
- Tablolar arasındaki ilişkileri oluşturunuz.
- Form tasarımlarını oluşturunuz.
- Formlar üzerinde ürün, müşteri ve siparişler ile ilgili ekleme, silme, güncelleme, listeleme ve arama işlemlerini yapınız.
- Bilgileri ekleyerek, geliştirdiğiniz uygulamanın çalışıp çalışmadığını test ediniz.
- Hata yönetimi yapınız. Program, istenen dışında hata mesajı göstererek kapanmamalıdır.
- Uygulamanın kurulum dosyasını oluşturunuz ve farklı bir bilgisayara kurulumunu yapınız.

6.9. ORM YAPISI VE ENTITY FRAMEWORK

Entity Framework veri tabanı işlemlerinin kod editöründe yapılmasına olanak sağlayan bir **ORM** (Object Relational Mapping) aracıdır. Bu araç ile veri tabanı tabloları ve programlama dilindeki sınıflar eşleştirilerek veri tabanı işlemlerinin uygulama üzerinden yapılması sağlanır. Bir başka deyişle veri tabanındaki her bir tablo sınıf olarak, tablolardaki her bir alan da sınıfa ait bir özellik olarak kullanılır. Bundan dolayı kullanıcıların SQL sorguları yerine programlama dilleri (C#, VB) ile veri tabanı işlemlerini yapması sağlanır. Entity Framework ile oluşturulan sınıflar ve metotlar kullanılarak veri tabanı tablolarında ekleme, silme, güncelleme ve listeleme işlemleri yapılabilir. Ayrıca **LINQ** (Language Integrated Query- Dile Entegre Edilmiş Sorgu) ile karmaşık sorgulama işlemleri de kolaylıkla yapılabilir.



Görsel 6.150: Entity Framework şeması

Entity Framework'ün temel amacı, uygulama geliştiricinin SQL sorgularına gerek kalmadan program içinde bütün veri tabanı işlemlerini yapabilmesidir. Örneğin klasik ADO.NET uygulamalarında bir bağlantının açılmasından kapatılmasına ve sorguların çalıştırılmasına kadar bütün işlemler geliştirici tarafından yapılır. Entity Framework kullanıldığında ise bu tür işlemler otomatik olarak bu araç tarafından yapılır. Entity Framework, kullanım olarak çok kolay fakat performans olarak ADO.NET teknolojisinden yavaştır.

Entity Framework ORM aracının üç farklı kullanımı şu şekildedir:

1. Database First

Database First yöntemi, DBMS programı üzerinde bir veri tabanı oluşturulup projede kullanılmak istenirse tercih edilir. Kullanıcı tarafından ilk olarak veri tabanı oluşturulur. Daha sonra projeye **ADO.NET Entity Data Model** eklenir ve **EF Designer from database** seçeneği ile işlem yapılmak istenen tablolar seçilerek **ORM** işlemi gerçekleştirilir. Tablolar sınıf olarak, alanlar da sınıfa ait özellik olarak görüntüle-

nerek veri tabanı işlemleri için kullanılabilir. Kısaca açıklamak gerekirse önce veri tabanı tabloları oluşturulur sonra Entity Framework vasıtasıyla sınıflar otomatik olarak oluşturulur.

2. Model First

Model First yöntemi, veri tabanı tablolarının görsel olarak kod editörü üzerinde oluşturulması için kullanılır. Projeye **ADO.NET Entity Data Model** eklenir ve **Empty EF Designer Model** seçeneği ile varlıkların oluşturulacağı ekran görüntülenir. Burada oluşturulan **Entity** adı verilen nesneler, veri tabanına tablo olarak eklenir. Aynı zamanda bu tablolarla eşleştirilen sınıflar da kod tarafında oluşturulur. Kısaca açıklamak gerekirse önce model oluşturulur sonra Entity Framework vasıtasıyla veri tabanı tabloları ve sınıflar otomatik olarak oluşturulur.

3. Code First

Code First yaklaşımında ilk olarak kod editöründe tablo ve tablonun alanlarına karşılık gelen sınıf ve özellikler yazılır. Daha sonra **Migration** yöntemi ile değişiklikler veri tabanına aktarılır. Database First yaklaşımının tam tersidir. Code First yaklaşımında veri tabanı tablolarındaki kısıtlamaların belirtilebilmesi için "Data Annotations" veya "Fluent API" yöntemleri kullanılabilir. Tablolar arası ilişkiler de bu yöntemlerle yapılır. Kısaca açıklamak gerekirse önce sınıflar oluşturulur sonra Entity Framework vasıtasıyla veri tabanı tabloları otomatik olarak oluşturulur.

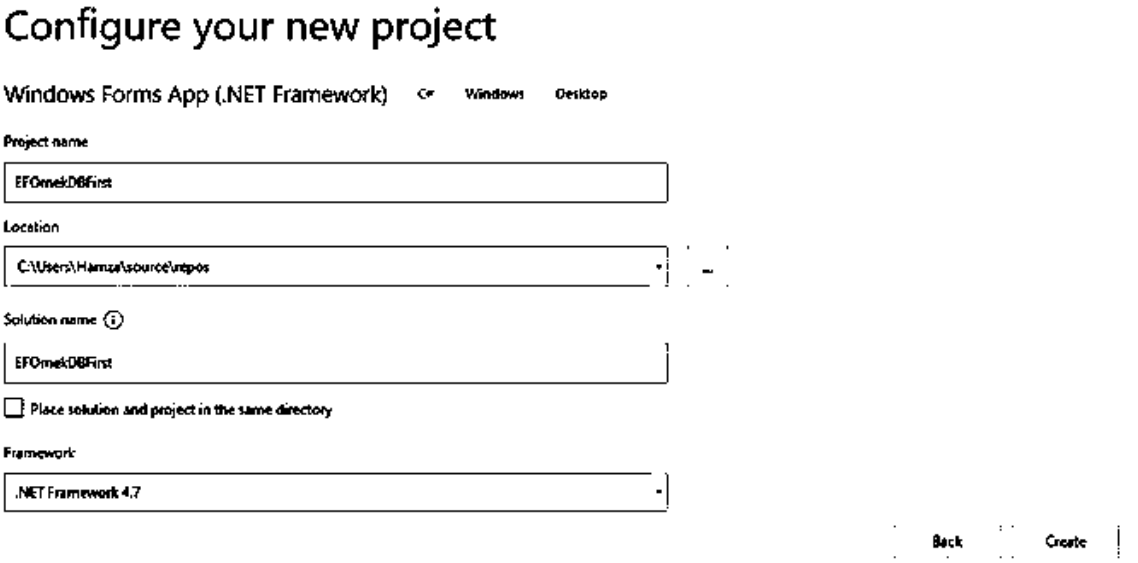
Sıra Sizde

Entity Framework ORM aracının kullanıldığı örnek uygulamaları inceleyiniz.

Entity Framework Database First yaklaşımı ile proje geliştirme

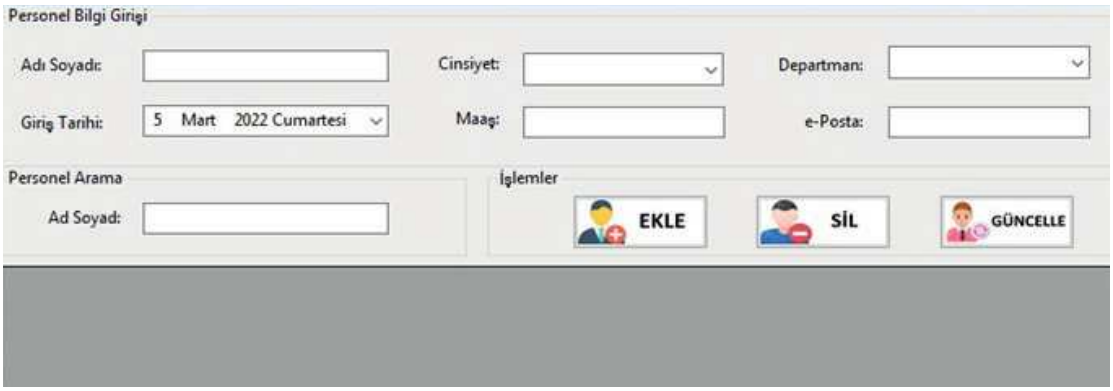
Daha önce hazırlanan sirketdb veri tabanı kullanılarak personel tablosu üzerinde CRUD (Create-Read-Update-Delete) işlemleri gerçekleştirecek bir form uygulaması oluşturulacaktır. Var olan bir veri tabanı kullanılacağı için bu projede Database First kullanmak daha doğru olur.

İlk olarak Görsel 6.151'de görüldüğü gibi "EFOrnekDBFirst" adında bir form projesi oluşturulur.



Görsel 6.151: Entity Framework projesi oluşturma

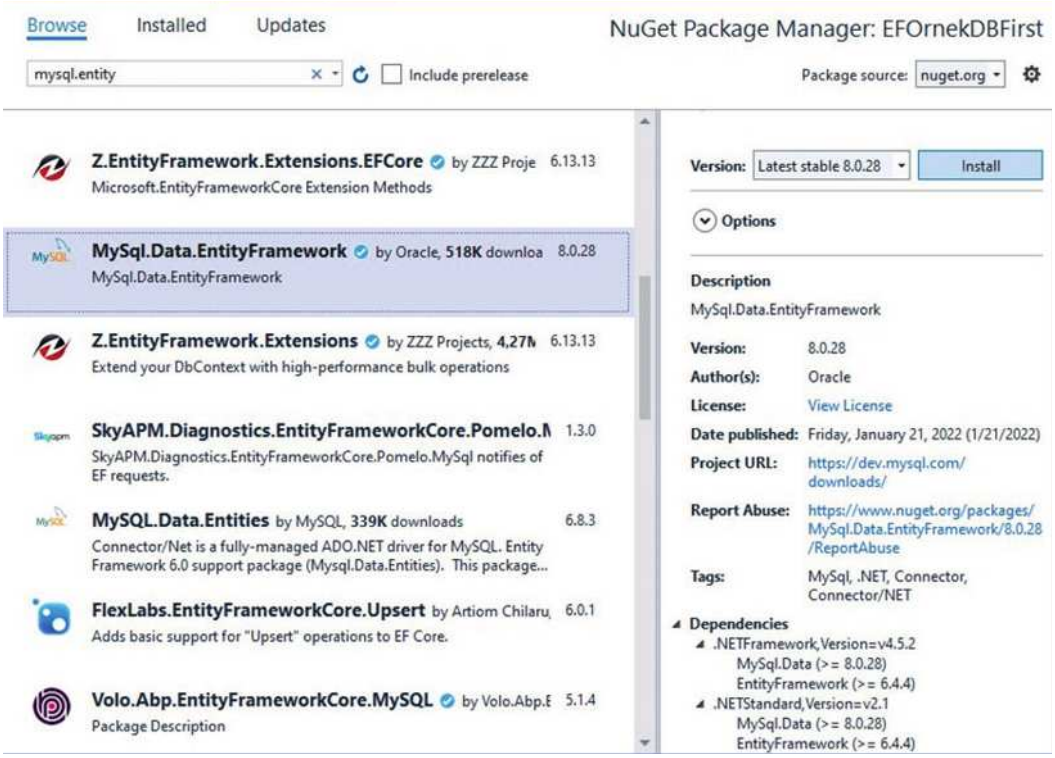
Oluşturulan form üzerinde personel tablosundaki verilere göre bir tasarım yapılır (Görsel 6.152). Personel veri tabanında personel_id, ad_soyad, cinsiyet, departman, giriş_tarihi, maas, eposta alanları bulunur. Form elemanlarının isimleri txtAdSoyad, comboCinsiyet, comboDepartman, dateTimeGiris, txtEposta, txtMaas, txtPersonelAra, btnEkle, btnSil, btnGuncelle ve dataGridPersonel şeklinde belirtilir.



Görsel 6.152: Personel İşlemleri form tasarımı

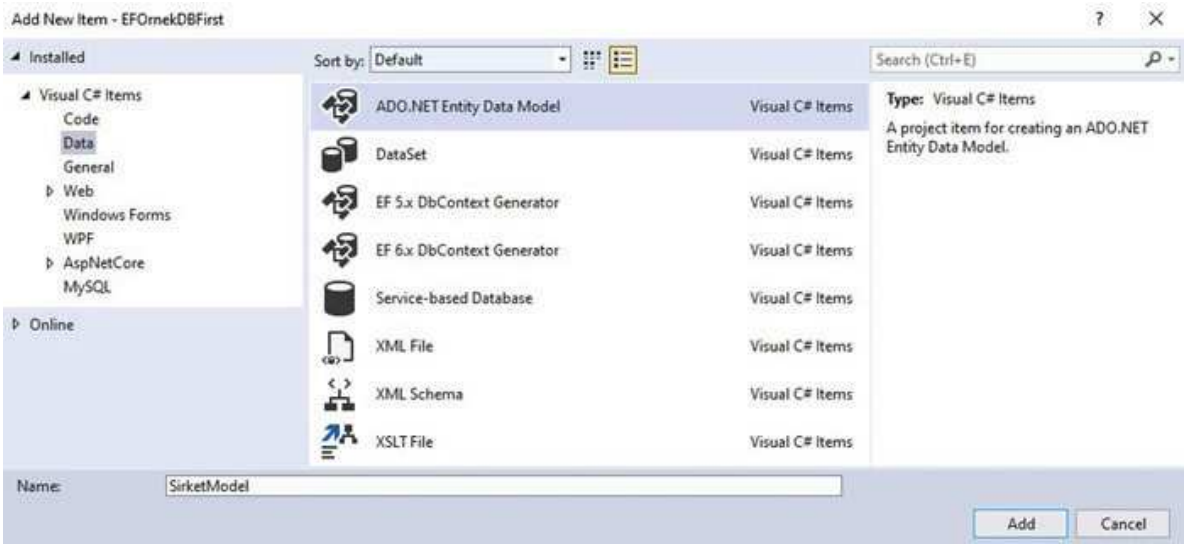
ComboBox nesnelerine veri ekleme işlemleri yapılmalıdır. Cinsiyet için "Kız ve Erkek", departman için "Bilişim Teknolojileri, Üretim, Satış, İnsan Kaynakları, Pazarlama, Güvenlik, Temizlik" değerleri eklenmelidir.

Entity Framework aracının MySQL veri tabanı ile birlikte projede kullanılabilmesi için paket yükleme işlemlerinin yapılması gerekir. Paketleri yüklemek için Solution Explorer penceresinde proje üzerine sağ tıklanarak "Manage NuGet Packages" seçeneği seçilir. Daha sonra gelen pencerede Entity Framework kullanımı için gerekli olan "MySql.Data" ve "EntityFramework" paketlerini içeren **MySql.Data.EntityFramework** paketi yüklenir (Görsel 6.153). Paketler ayrı ayrı da yüklenebilir.



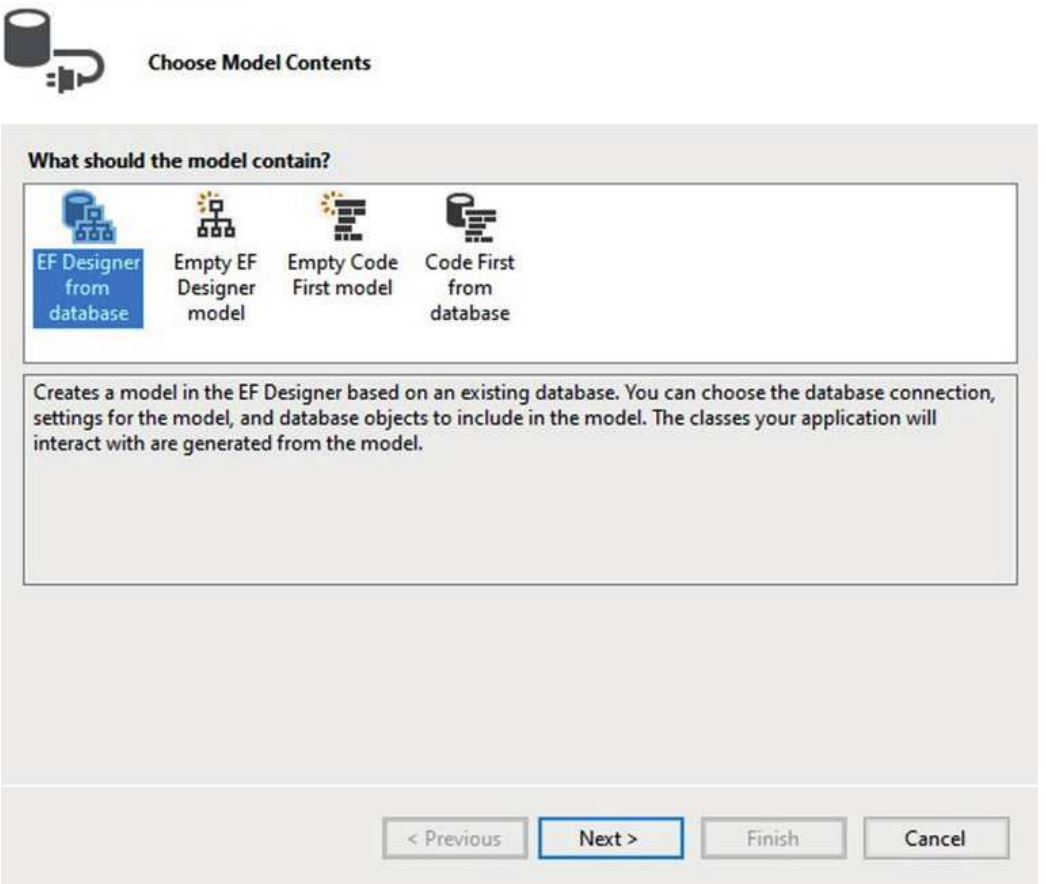
Görsel 6.153: Entity Framework ve MySQL kullanımı için gerekli olan paketlerin yüklenmesi

Proje üzerine sağ tıklanarak Add > New Item seçeneği seçilir. Gelen pencerede **ADO.NET Entity Data Model** seçilir. "SirketModel" adı verilerek Add butonuna tıklanır (Görsel 6.154).



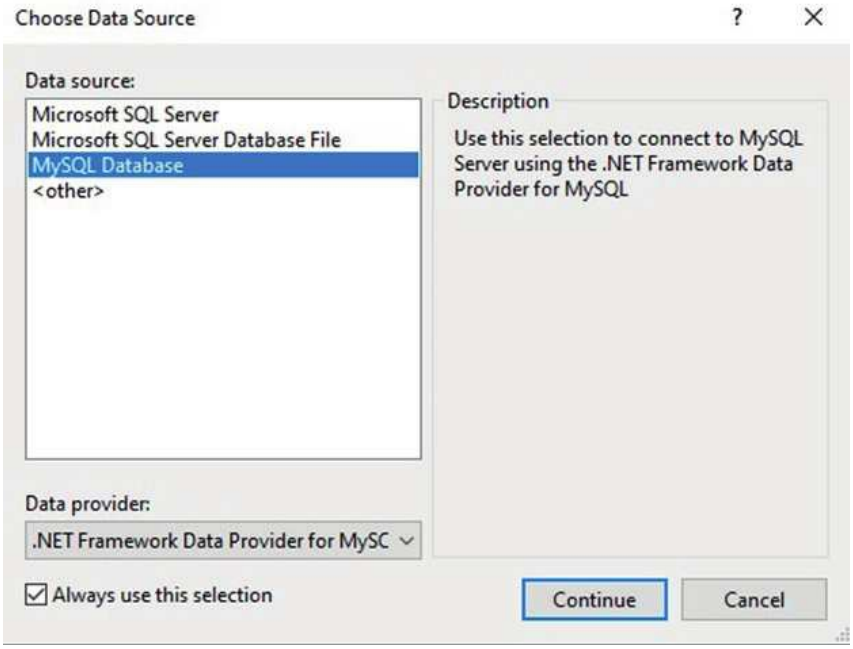
Görsel 6.154: Projeye ADO.NET Entity Data Model eklenmesi

Gelen pencereden "EF Designer from database" seçilerek Next tuşu ile devam edilir (Görsel 6.155). Bu seçenek Database First yaklaşımının kullanılacağı, bir başka deyişle veri tabanında daha önce oluşturulan tablolarla işlem yapılacağı anlamına gelir.



Görsel 6.155: Kullanılacak Entity Framework yaklaşımının seçilmesi

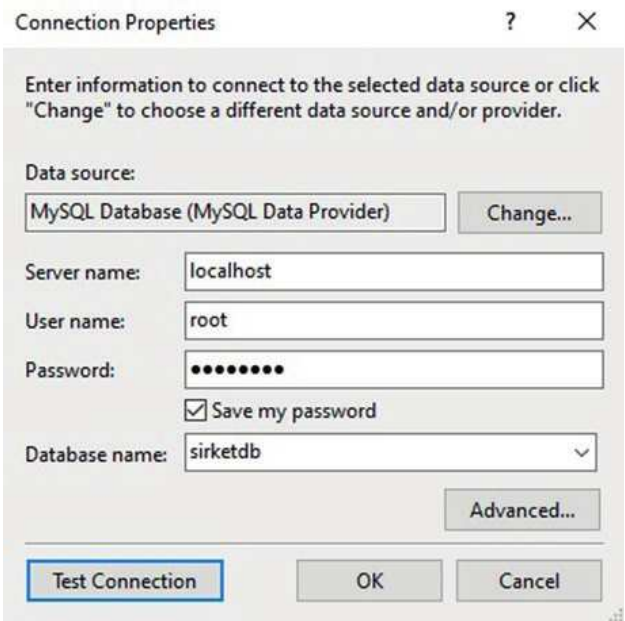
Bu aşamada daha önce oluşturulan bağlantılar görüntülenir. İlk kez bağlantı yapılacaksa veya başka bir bağlantı oluşturmak istenirse **New Connection** seçeneğine tıklanır. Gelen ekrandan veri kaynağı olarak **MySQL Database** seçilir ve Continue butonuna tıklanarak işleme devam edilir (Görsel 6.156).



Görsel 6.156: Veri kaynağının seçilmesi

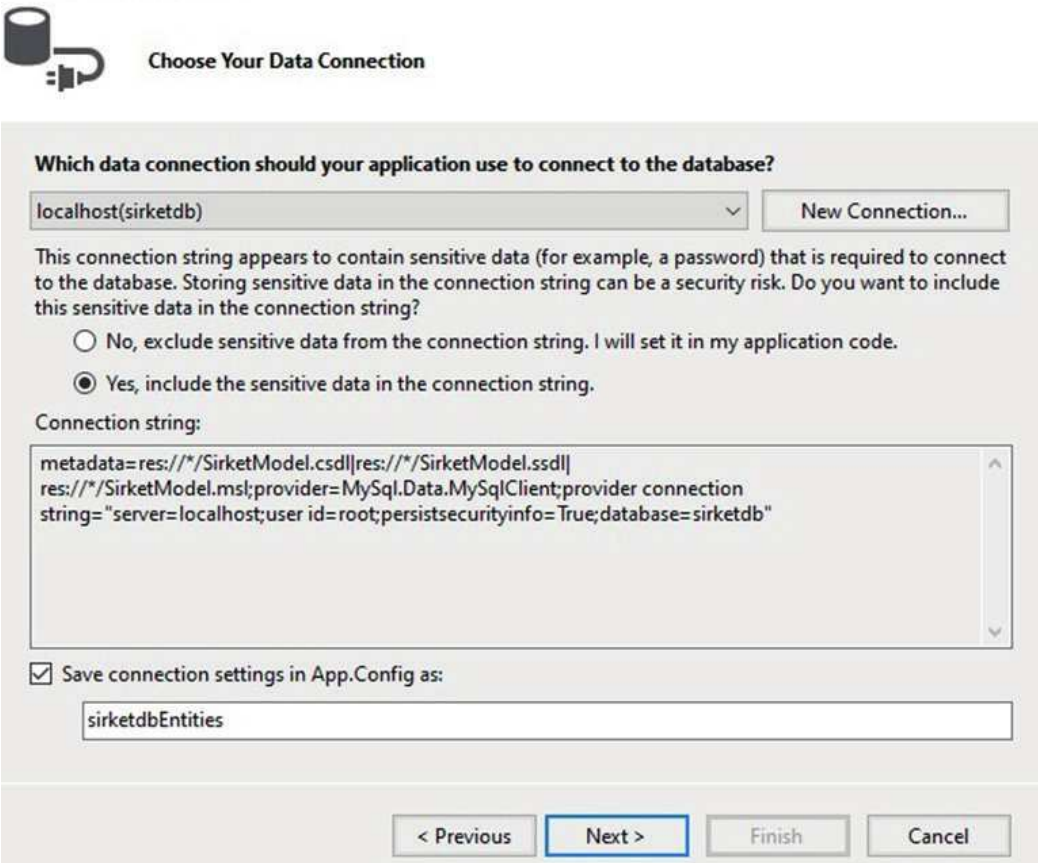
Not Bu aşamada "MySQL Database" seçeneği görünmezse kod editörüne bağlanma ile ilgili eksik kurulumların olduğu anlaşılır. **Connector/NET** ve **MySQL for Visual Studio** eklentilerinden eksik olanlar kurularak bağlantı sorunları giderilir.

Görsel 6.157'de sunucu ve veri tabanı ile ilgili bilgiler doğru bir şekilde girilmelidir. "Test Connection" butonu tıklanarak bağlantı test edilir. Bağlantı sağlandıktan sonra OK butonuna tıklanarak bağlantı ekranına geri dönülür.



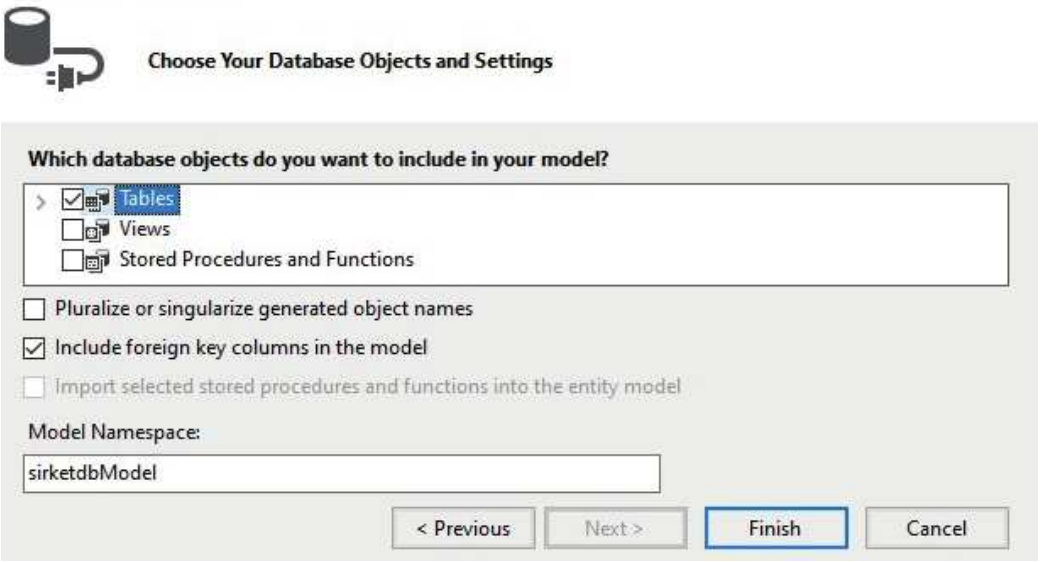
Görsel 6.157: Bağlantı özelliklerinin belirtilmesi

Bağlantı seçim ekranında oluşturulan bağlantı seçili olarak görüntülenir (Görsel 6.158). Alt bölümde bağlantı ayarlarının "sirketdbEntities" adıyla **app.Config** dosyasının içine kaydedileceği belirtilir. İstenirse bu isim değiştirilebilir.



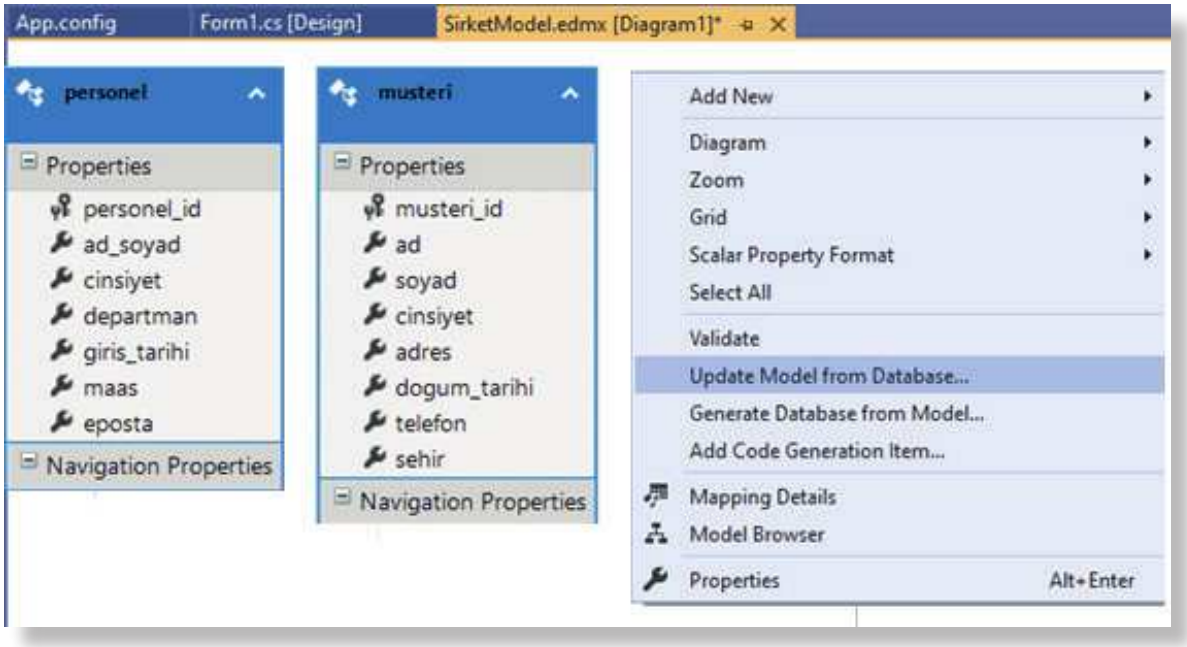
Görsel 6.158: Bağlantı seçim ekranı

Veri tabanı bağlantısı yapıldıktan sonra oluşturulacak modele hangi veri tabanı nesnelerinin dâhil edileceği seçilir (Görsel 6.159). Veri tabanında diğer nesneler (görünüm, fonksiyon ve saklı yordam) oluşturulmadığı için yalnızca Tables işaretlenir ve Finish butonu tıklanarak işlem bitirilir.



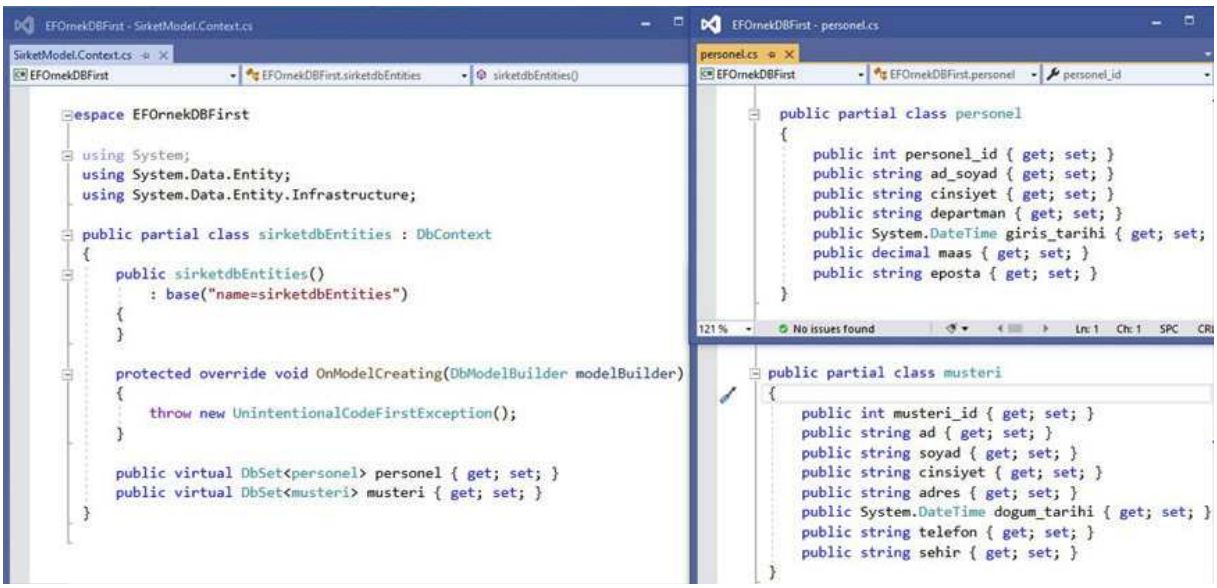
Görsel 6.159: Modele eklenecek veri tabanı nesnelerinin seçimi

Bu işlemden sonra model ve sınıfların oluşması için bir süre beklenir. Model, **SirketModel.edmx** dosyasında Diagram şeklinde görüntülenir (Görsel 6.160). Veri tabanı üzerinde değişiklik yapıldığı zaman bu değişikliğin programda da uygulanması istenirse model üzerinde sağ tıklanarak "Update Model from Database" seçeneği seçilir. Bu değişikliğin sınıflara da yansıması için proje tekrar derlenir.



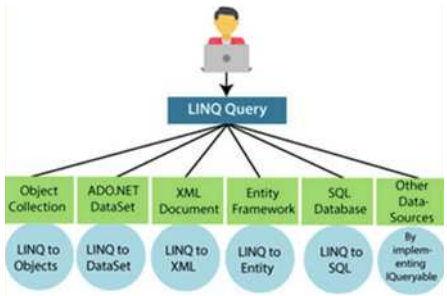
Görsel 6.160: Oluşturulan modelin görüntülenmesi

Solution Explorer penceresinde modelin içinde model ile birlikte sınıfların da oluştuğu görülür. Bunlar, her tablo için tablo ile aynı özellikleri içeren bir sınıf ve bu sınıfları kullanarak veri tabanı işlemlerini yapacak Context sınıfıdır (Görsel 6.161).



Görsel 6.161: Tablolara karşılık gelen sınıflar ve Context sınıfı

LINQ (Language Integrated Query), dile entegre edilmiş sorgu olarak Türkçeye çevrilebilir. LINQ ile C# ve VB.NET programlama dilleri kullanılarak farklı kaynaklardaki veri setleri üzerinde CRUD (Create - Read - Update - Delete) işlemleri yapılabilir. LINQ genel olarak koleksiyonlar, Data Set, XML, Web Servisler, SQL veri tabanı, Entity Framework gibi içinde verileri barındıran nesnelerde sorgulama işleminin yapılması için kullanılır (Görsel 6.162). LINQ kullanımı sayesinde özellikle karmaşık veri setlerinde sorgulama yapılırken büyük kolaylık sağlanır.



Görsel 6.162: LINQ şeması

LINQ kullanımı, kütüphane veri tabanındaki tabloları birleştirme işlemi ile gösterilmiştir (Görsel 6.163). SQL sorgusu ile benzer özellikler taşısa da editörün kod tamamlama özelliği sayesinde LINQ kullanımı daha kolaydır. Sorgu incelendiğinde tabloların join komutu ile ilişkili alanlar eşitlenerek birleştirildiği görülür. Daha sonra yeni bir nesne oluşturulup veriler istenen şekilde listelenir.

```
private void Form1_Load(object sender, EventArgs e)
{
    kutuphaneEntities kutuphaneDb = new kutuphaneEntities();
    var sonuc = from o in kutuphaneDb.odunc_kitaplar
                join k in kutuphaneDb.kitaplar on o.kitap_id equals k.kitap_id
                join t in kutuphaneDb.kitap_turleri on k.tur_id equals t.tur_id
                join ogr in kutuphaneDb.ogrenciler on o.ogr_no equals ogr.ogrenci_no
                select new
                {
                    islemID = o.id,
                    OgrenciNo = ogr.ogrenci_no,
                    OgrenciAdi = ogr.ad,
                    OgrenciSoyadi = ogr.soyad,
                    KitapTuru = t.tur_adi,
                    KitapAdi = k.kitap_adi,
                    Yazari = k.yazar,
                    VerilmeTarihi = o.verilis_tarihi,
                    TeslimTarihi = o.teslim_tarihi
                };
    dataGridOduncKitaplar.DataSource = sonuc.ToList();
}
```

Görsel 6.163: LINQ kullanımı

Görsel 6.164'te LINQ sorgusu sonucunda verilerin yeni oluşturulan alan adları ile listelendiği görülür.

Form1

islemID	OgrenciNo	OgrenciAdi	OgrenciSoyadi	KitapTuru	KitapAdi	Yazari	VerilmeTarihi	TeslimTarihi
11	145	Esat	E.	Roman	Beyaz Gemi	Cengiz Aytma...	9.02.2022	
12	222	Zeynep	Ö.	Roman	Suç ve Ceza	Dostoyevski	9.02.2022	5.03.2022
13	336	Murat	T	Şiir	Safahat	Mehmet Akif ...	9.02.2022	
14	555	Ayşe	C.	Şiir	Otuz Beş Yaş	Cahit Sıtkı Ta...	16.02.2022	5.03.2022
15	985	Mehmet	D.	Roman	Kuyucaklı Yusuf	Sabahattin Ali	16.02.2022	
16	411	Samet	K.	Roman	Sefiller	Victor Hugo	16.02.2022	
17	99	Ayşe	Y.	Hikaye	Ömer Seyfettin H...	Ömer Seyfettin	18.02.2022	
18	150	Emirhan	Ç.	Roman	Ölü Canlar	Gogol	24.02.2022	
19	344	Esra	Ö.	Roman	Küçük Ağa	Tarık Buğra	18.02.2022	

Görsel 6.164: LINQ sorgusu sonucunda listelenen veriler

"Lambda Expression" => Lambda ifadesi, veriler üzerinde işlem yapılırken değer atama ve filtreleme işlemlerinde kolaylık sağlayan isimsiz fonksiyondur. Entity Framework'te listeleme işlemlerinde koşul belirtilirken sıklıkla kullanılır. Lambda ifadesinin kullanımı şu şekildedir:

sirketDb.personel.FirstOrDefault (p => p.personel_id == 7)

Örnek kullanımda parantez içindeki koşul belirtilen bölüm bir Lambda ifadesidir. Burada p harfi personel nesnesini temsil eder. Kısa bir kod bloku ile personel_id değeri 7 olan personel listelenir.

Entity Framework'le CRUD İşlemleri

1. Veri Listeleme

Personel İşlemleri sayfası açıldığında verilerin listelenmesi için Görsel 6.165'teki kodların yazılması yeterlidir. Öncelikle **sirketdbEntities** sınıfının nesne örneği oluşturulur. Oluşturulan bu nesne ile sirketdbEntities sınıfında DbSet olarak tanımlanan tabloların verilerine kolay bir şekilde erişildiği görülür. Personel tablosunun içeriği liste hâline dönüştürülerek DataGridView nesnesine aktarılır.

```
sirketdbEntities sirketDb;
private void Form1_Load(object sender, EventArgs e)
{
    try
    {
        sirketDb = new sirketdbEntities();
        dataGridPersonel.DataSource = sirketDb.personel.ToList();

    }
    catch (Exception)
    {

        MessageBox.Show("Listeleme işleminde bir hata oluştu");
    }

}
```

Görsel 6.165: Entity Framework'te veri listeleme

Sıra Sizde

Personellerin listelendiği DataGridView nesnesindeki sütunların başlıklarını değiştiriniz.

2. Veri Ekleme

Personel İşlemleri sayfasına veri eklemek için öncelikle kaydedilecek personel nesnesinin oluşturulması ve alacağı değerlerin belirtilmesi gerekir. Daha sonra ekleme işlemi gerçekleştirilir (Görsel 6.166). Kaydetme komutları butonun tıklanma olayına yazılır. Eğer sirketDb nesnesi üzerinde yapılan değişiklikler kaydedilmezse bu işlem veri tabanına uygulanmaz. Bir başka deyişle veri kaydedilmez.

```
private void btnEkle_Click(object sender, EventArgs e)
{
    try
    {
    // Eklenecek olan personel oluşturulur.
        personel personel = new personel();
        personel.ad_soyad = txtAdSoyad.Text;
        personel.cinsiyet = comboCinsiyet.Text;
        personel.departman = comboDepartman.Text;
        personel.giris_tarihi = dateTimeGiris.Value.Date;
        personel.eposta = txtEposta.Text;
        personel.maas = Convert.ToDecimal(txtMaas.Text);
    // Oluşturulan personel tabloya eklenir
        sirketDb.personel.Add(personel);
    // Değişiklik kayıt edilir.
        sirketDb.SaveChanges();
// Eklenen kaydın görülebilmesi için veriler DataGridView'de tekrar listelenir
        dataGridPersonel.DataSource = sirketDb.personel.ToList();
    }
    catch (Exception)
    {
        MessageBox.Show("Kayıt işleminde bir hata oluştu");
    }
}
```

Görsel 6.166: Entity Framework'te veri listeleme

Sıra Sizde

1. Form elemanlarının içeriğini temizleyen bir metot hazırlayınız. Kayıt ekleme ve güncelleme işlemlerinden sonra gerekli yerde bu metodu çağırınız.
2. Personel İşlemleri form ekranı üzerinden 10 adet personel kaydı yapınız.

3. Veri Silme

Veri silme işlemi yapılmadan önce silinmesi istenen personel, DataGridView üzerinde seçili hâle getirilerek form elemanlarında gösterilmelidir (Görsel 6.167).

```
private void dataGridPersonel_CellClick(object sender, DataGridViewCellEventArgs e)
{
    try
    {
        txtAdSoyad.Text = dataGridPersonel.CurrentRow.Cells["ad_soyad"].Value.ToString();
        comboCinsiyet.Text = dataGridPersonel.CurrentRow.Cells["cinsiyet"].Value.ToString();
        comboDepartman.Text = dataGridPersonel.CurrentRow.Cells["departman"].Value.ToString();
        txtEposta.Text = dataGridPersonel.CurrentRow.Cells["eposta"].Value.ToString();
        txtMaas.Text = dataGridPersonel.CurrentRow.Cells["maas"].Value.ToString();
        dateTimeGiris.Value = DateTime.Parse((dataGridPersonel.CurrentRow.Cells["giris_tarihi"].Value.ToString()));
    }
    catch (Exception)
    {

        MessageBox.Show("Hata oluştu");
    }

}
```

Görsel 6.167: Silinecek verinin seçili hâle getirilmesi

DataGridView nesnesinde seçilen kişinin bilgileri form elemanlarına aktarıldıktan sonra silme veya güncelleme işlemi yapılabilir. Sil butonu tıklandığında personelin silinmesini sağlayan kodlar Görsel 6.168'de görülür.

```
private void btnSil_Click(object sender, EventArgs e)
{
    try
    {
        // İlk olarak seçili olan personelin personel_id değeri alınır
        int id= Int32.Parse(dataGridPersonel.CurrentRow.Cells["personel_id"].Value.ToString());
        // personel_id değeri kullanılarak silinecek olan personel bulunur.
        personel silinecekPersonel = sirketDb.personel.FirstOrDefault(p => p.personel_id == id);
        // personel tablodan silinir
        sirketDb.personel.Remove(silinecekPersonel);
        // Değişikler kayıt edilir.
        sirketDb.SaveChanges();
        // Silinen kaydın görünmemesi için DataGridView nesnesi yeniden doldurulur
        dataGridPersonel.DataSource = sirketDb.personel.ToList();
    }
    catch (Exception)
    {
        MessageBox.Show("Silme işleminde hata oluştu");

    }
}
```

Görsel 6.168: Entity Framework'te veri silme

Sıra Sizde

Personel İşlemleri form ekranı üzerinden üç personelin kaydını siliniz.

4. Veri Güncelleme

Güncellenecek personel DataGridView üzerinde seçildikten sonra personel ile ilgili bilgiler form elemanlarında gösterilir. Yeni değerler girildikten sonra Güncelle butonuna tıklanarak güncelleme işlemi yapılır. Güncelleme işlemi için gerekli olan kodlar Görsel 6.169'da görülür.

```
private void btnGuncelle_Click(object sender, EventArgs e)
{
    try
    {   // İlk olarak seçili olan personelin personel_id değeri alınır
        int id = Int32.Parse(dataGridPersonel.CurrentRow.Cells["personel_id"].Value.ToString());
        // personel_id değeri kullanılarak güncellenecek olan personel bulunur.
        personel guncellenecekPersonel = sirketDb.personel.FirstOrDefault(p => p.personel_id == id);
        // Yeni değerler seçilen nesnenin alanlarına aktarılır
        guncellenecekPersonel.ad_soyad = txtAdSoyad.Text;
        guncellenecekPersonel.cinsiyet = comboCinsiyet.Text;
        guncellenecekPersonel.departman = comboDepartman.Text;
        guncellenecekPersonel.giris_tarihi = dateTimeGiris.Value.Date;
        guncellenecekPersonel.eposta = txtEposta.Text;
        guncellenecekPersonel.maas = Convert.ToDecimal(txtMaas.Text);
        sirketDb.SaveChanges(); // Değişiklikler kaydedilir
        dataGridPersonel.DataSource = sirketDb.personel.ToList();
        Temizle();
    }
    catch (Exception)
    {
        MessageBox.Show("Güncelleme işleminde hata oluştu");
    }
}
```

Görsel 6.169: Entity Framework'te veri güncelleme

5. Arama İşlemi

Personellerin içinde "ad_soyad" alanına göre arama işleminin yapılması için Görsel 6.170'teki kodlar yazılır. Arama kutusundaki her değişiklikte bu kodlar tekrar çalışır ve veriler yeniden listelenir.

```
private void txtPersonelAra_TextChanged(object sender, EventArgs e)
{
    try
    {
        var sonuc = from p in sirketDb.personel
                    //where p.ad_soyad.Contains(txtPersonelAra.Text) // İçeren değerleri listeler
                    where p.ad_soyad.StartsWith(txtPersonelAra.Text) // Başlayan değerleri listeler
                    select p;

        dataGridPersonel.DataSource = sonuc.ToList();
    }
    catch (Exception)
    {
        MessageBox.Show("Arama işleminde hata oluştu");
    }
}
```

Görsel 6.170: Entity Framework'te arama işlemi

Sıra Sizde

1. Müşteri İşlemleri adında yeni bir form tasarlayarak "musteriler" tablosu için listeleme, ekleme, silme, güncelleme ve arama işlemlerini Entity Framework kullanarak gerçekleştiriniz.
2. Aşağıda listelenen şartlara göre Windows form ile Telefon Rehberi uygulaması yapınız.
   - Uygulamada Entity Framework aracı kullanılmalıdır.
   - Rehbere kişi kaydetme, silme, güncelleme ve kişileri listeleme işlemleri yapılmalıdır.
   - Rehberde kişi adına göre arama işlemi yapılmalıdır.
   - Kişileri alfabetik sıraya göre sıralama işlemi yapılmalıdır.
   - Proje tamamlandıktan sonra kurulum dosyası oluşturularak farklı bir bilgisayara kurulmalıdır.

ÖLÇME VE DEĞERLENDİRME

A) Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

1. Aşağıdakilerden hangisi bir Veri Tabanı Yönetim Sistemi değildir?

A) MySQL B) Oracle C) SQL Server
D) SQL E) IBM DB2

2. Bağlantı cümlesi oluşturulurken aşağıdakilerden hangisi belirtilmez?

A) Şifre B) Sunucu C) Veri tabanı
D) Tablo E) Kullanıcı adı

3. Aşağıdakilerden hangisi veri tabanı tablolarındaki her satır için tanımlayıcı görevi gören, benzersiz ve boş bırakılamaz bir kısıtlama olan anahtardır?

A) Foreing Key B) Primary Key C) İndex
D) Unique Key E) Auto_increment

4. Aşağıdakilerden hangisi oluşturulan bir veri tabanı veya tablonun silinmesi için kullanılan komuttur?

A) Delete B) Remove C) Clear
D) Alter E) Drop

5. Aşağıdakilerden hangisi tekrarlı veriler içeren sütunlarda farklı değerlerin listelenmesini sağlayan komuttur?

A) DISTINCT B) IN C) SUM
D) AVG E) COUNT

6. Aşağıdakilerden hangisi veri tabanı tablolarında sıralama işleminin yapılması için kullanılan anahtar kelimedir?

A) WHERE B) INNER JOIN C) Order By
D) Group By E) DISTINCT

7. Aşağıdakilerden hangisi Musteriler tablosundaki 10 numara olarak kayıtlı bir müşterinin şehir bilgisini güncelleyen SQL sorgusudur?

A) update Musteriler set sehir=”Samsun”
B) update Musteriler where sehir=”Samsun”
C) update Musteriler where sehir=”Samsun” and musteri_no=10
D) update Musteriler set sehir=”Samsun” and musteri_no=10
E) update Musteriler set sehir=”Samsun” where musteri_no=10

8. Aşağıdakilerden hangisi MySQL veri tabanının sunucuya dâhil edilmesi için kullanılır?

A) Data Export B) New Query C) Server Status
D) Dashboard E) Data Import

9. Aşağıdakilerden hangisi veri tabanına bağlantı kurmak için kullanılan sınıftır?

A) MySqlDataReader B) MySqlConnection C) MySqlCommand
D) MySqlDataAdapter E) DataSet

10. Çalıştırılmak istenen sorgu, MySqlCommand nesnesinin hangi parametresinde belirtilmelidir?

A) Connection B) DataSource C) ConnectionString
D) CommandText E) Parameters

11. Aşağıdakilerden hangisi DataGridView nesnesinin veri kaynağının belirtildiği özelliktir?

A) MultiSelect B) AutoSizeColumnsMode C) DataSource
D) HeaderText E) Name

12. Aşağıdakilerden hangisi veri tabanlarında çok fazla sütun ve satırdan oluşan bir tabloyu tekrarlardan arındırmak için daha az satır ve sütun içeren alt kümelere ayrıştırma işlemidir?

A) Normalizasyon B) İlişkisel veri tabanı C) Veri tabanı tasarımı
D) İyileştirme E) Tablo oluşturma

13. Aşağıdakilerin hangisi Entity Framework'te ilk olarak kodlarla sınıfların yazılıp daha sonra bu sınıflara göre veri tabanında tabloların oluşmasını sağlayan yaklaşımdır?

A) Database First B) Code First C) ADO.NET
D) Model First E) MySQL

GENEL AĞ KAYNAKÇASI

- https://bidb.itu.edu.tr Erişim Tarihi: 22.03.2022 Saat: 09.00
- https://www.tubitak.gov.tr Erişim Tarihi: 22.03.2022 Saat: 09.10
- https://www.btkakademi.gov.tr Erişim Tarihi: 22.03.2022 Saat: 09.15
- https://sozluk.gov.tr Erişim Tarihi: 22.03.2022 Saat: 09.20
- https://www.tdk.gov.tr Erişim Tarihi: 25.03.2022 Saat: 09.25

GÖRSEL KAYNAKÇA

http://kitap.eba.gov.tr/karekod/Kaynak.php?-KOD=1537

CEVAP ANAHTARLARI

1. ÖĞRENME BİRİMİ

A

1 Yanlış	**2** Doğru	**3** Doğru	**4** Yanlış	**5** Yanlış
6 Doğru	**7** Yanlış			

B

8 using	**9** ToolBox veya Araç Kutusu	**10** %	**11** ToString()	**12** Kaydet namespace

13

- Değişken isimlerinde boşluk karakteri kullanılmaz.
- Değişken isimleri sayı ile başlamaz.
- Doğru tanımlanmıştır.
- Doğru tanımlanmıştır.
- Doğru tanımlanmıştır.
- Ondalık sayılar int veri türünde tanımlanamaz.
- char olarak tanımlı değişkenlere tek tırnak içine alınmış sadece bir karakter atanabilir.
- Doğru tanımlanmıştır.

14

3,2 sayısı mesaj olarak görülür.

15

```
private void button1_Click(object sender, EventArgs e)
{
int sayi;
double sonuc;
sayi = Convert.ToInt32(textBox1.Text);
sonuc = sayi * 0.18;
MessageBox.Show(sonuc.ToString());
}
```

16

```
private void button1_Click(object sender, EventArgs e)
{
int yariCap;
double alan,cevre,piSayisi;
piSayisi = 3.14;
yariCap = Convert.ToInt32(textBox1.Text);
alan = piSayisi * yariCap * yariCap;
cevre = 2 * piSayisi * yariCap;
MessageBox.Show("Dairenin alanı=" + alan.ToString());
MessageBox.Show("Dairenin çevresi=" + cevre.ToString());
}
```

2. ÖĞRENME BİRİMİ

A	**1** Yanlış	**2** Yanlış	**3** Doğru	**4** Doğru	**5** Yanlış

B	**6** if(a > b && a > c)	**7** if(a < b || a < c)	**8** 3 – 6 – 9 – 12 15 – 17 – 18	**9** 1 – 2 – 4 – 5 7 – 8 – 10 – 11 – 13 – 14

10

- 1. tur sonuç=1
- 2. tur sonuç=2
- 3. tur sonuç=6
- 4. tur sonuç=24
- 5. tur sonuç=120
- 6. tur sonuç=720

11

Kodlar çalıştırılmaz ve program başlamaz.

Açıklama: Kodlar derlenirken Hata Listesi panelinde "string türü örtülü olarak int türüne dönüştürülemez."

mesajı verir ve kodlar çalıştırılmaz çünkü yapılan hata, derleme hatasıdır.

Bu tip hatalar, programın başlatılmasını engelleyen çok kritik hatalardır.

12

```
private void button1_Click_1(object sender, EventArgs e)
{
int sayi1, sayi2, sayi3;
sayi1 = Convert.ToInt32(textBox1.Text);
sayi2 = Convert.ToInt32(textBox2.Text);
sayi3 = Convert.ToInt32(textBox3.Text);
if(sayi1>sayi2 && sayi1 > sayi3)
{
MessageBox.Show("İlk girilen sayı en büyüktür.");
}
if(sayi2>sayi1 && sayi2 > sayi3)
{
MessageBox.Show("İkinci girilen sayı en büyüktür.");
}
if (sayi3 > sayi1 && sayi3 > sayi2)
{
MessageBox.Show("Son girilen sayı en büyüktür.");
}
}
```

A

1

```
class Televizyon
{
private int sesSeviyesi;
private double ekranBoyutu;
private string goruntuTeknolojisi;
public int SesSeviyesi
{
get
{
return sesSeviyesi;
}
set
{
sesSeviyesi = value;
}
}
public double EkranBoyutu
{
get
{
return ekranBoyutu;
}
set
{
ekranBoyutu = value;
}
}
public string GoruntuTeknolojisi
{
get
{
return goruntuTeknolojisi;
}
set
{
goruntuTeknolojisi = value;
}
}
}
```

2

```
private class Bilgisayar
{
double ram;
public double RAMKapasitesi
{
get { return ram; }
set { ram = value; }
}
string cpu;
public string CPU
{
get { return cpu; }
set { cpu = value; }
}
double hd;
public double HDKapasitesi
{
get { return hd; }
set { hd = value; }
}
}
```

3

```
class Televizyon
{
public int SesSeviyesi { get; set; }
public double EkranBoyutu { get; set; }
public string GoruntuTeknolojisi { get; set;
}
}
```

4

```
class Bilgisayar
{
public double RAMKapasitesi { get; set; }
public string CPU { get; set; }
public double HDKapasitesi { get; set; }
}
```

5

```
class Televizyon
{
public int SesSeviyesi { get; set; }
public double EkranBoyutu { get; set; }
public string GoruntuTeknolojisi { get; set;
}
bool gucAcik = false;
int kanalNo = 1;
public void GucAc()
{
gucAcik = true;
}
public void GucKapat()
{
gucAcik = false;
}
public void KanalDegistir(int kanalNo)
{
this.kanalNo = kanalNo;
}
public int SesSeviyesiOku()
{
return SesSeviyesi;
}
}
```

A

6

```
class Televizyon
{
public int SesSeviyesi { get; set; }
public double EkranBoyutu { get; set; }
public string GoruntuTeknolojisi { get; set;
}
bool gucAcik = false;
int kanalNo = 1;
public int KanalNo
{
get { return kanalNo; }
}
public void GucAc()
{
gucAcik = true;
}
public void GucKapat()
{
gucAcik = false;
}
public void KanalDegistir(int kanalNo)
{
this.kanalNo = kanalNo;
}
public void KanalNoArtir()
{
kanalNo++;
}
public void KanalNoArtir(int artis)
{
kanalNo += artis;
}
public void KanalNoAzalt()
{
kanalNo--;
}
public void KanalNoAzalt(int azalis)
{
kanalNo -= azalis;
}
public int SesSeviyesiOku()
{
return SesSeviyesi;
}
}
private static void Main(string[] args)
{
Televizyon tv = new Televizyon();
tv.KanalDegistir(20);
System.Console.WriteLine(tv.KanalNo);
tv.KanalNoArtir();
System.Console.WriteLine(tv.KanalNo);
tv.KanalNoArtir(5);
System.Console.WriteLine(tv.KanalNo);
tv.KanalNoAzalt();
System.Console.WriteLine(tv.KanalNo);
tv.KanalNoAzalt(3);
System.Console.WriteLine(tv.KanalNo);
}
```

A

7

public: Sınıf ögelerine dışarıdan erişim sağlamak için kullanılır.

private: Bu tür ögelere sadece sınıf içinden erişilebilir.

8

```
abstract class Televizyon
{
protected int SesSeviyesi { get; set; }
public double EkranBoyutu { get; set; }
public string GoruntuTeknolojisi { get; set; }
bool gucAcik = false;
int kanalNo = 1;
public int KanalNo
{
get { return kanalNo; }
}
public abstract void GucAc();
public abstract void GucKapat();
public void KanalDegistir(int kanalNo)
{
this.kanalNo = kanalNo;
}
public void KanalNoArtir()
{
kanalNo++;
}
public void KanalNoArtir(int artis)
{
kanalNo += artis;
}
public void KanalNoAzalt()
{
kanalNo--;
}
public void KanalNoAzalt(int azalis)
{
kanalNo -= azalis;
}
public int SesSeviyesiOku()
{
return SesSeviyesi;
}
}
class AkilliTelevizyon : Televizyon
{
public string IsletimSistemi { get; set; }
public override void GucAc()
{
}
public override void GucKapat()
{
}
}
```

9

Televizyon sınıfına statik bir alan eklenir.

public static string Marka = "...";

10

Televizyon sınıfı sealed olarak tanımlanır.

sealed class Televizyon

11

Sınıf içindeki bir değişkeni, dış dünyaya kapatıp sadece sınıf içinde kullanılabilir kılmak için özellik private şeklinde tanımlanmalıdır.

12

Sınıf içindeki bir değişkeni, dış dünyaya kapatıp sadece sınıf içinde ve bu sınıftan türetilen alt sınıflarda kullanılabilir kılmak için özellik protected şeklinde tanımlanmalıdır.

13

Sınıf içindeki bir değişkeni her yerden erişilebilir kılmak için özellik public şeklinde tanımlanmalıdır

14

8

15

- Nesne oluşturulurken otomatik çalışan metot:
- Yapıcı metot (Constuctor)
- Sınıf adı ile aynı ada sahip olmalıdır.
- Soyut tanımlanmamalıdır.
- Sadece bir tane statik yapıcı metot olabilir.
- Dönüş tipi olamaz (void dâhil).
- Aşırı yüklenebilir.
- Nesne yok edilirken otomatik çalışan metot:
- Yıkıcı metot (Destructor)
- Sadece bir tane olabilir (Aşırı yüklenemez).
- Sınıf adı ile aynı ada sahip olmalıdır.
- Adı ~ (tilde) karakteri ile başlamalıdır.
- Dönüş tipi olamaz (void dâhil).

16

0 & 125

17

Sınıftan nesne oluşturmadan doğrudan sınıf adı ile sınıf ögelerine erişim için kullanılır.

18

Sınıfa ilk erişim sağlandığında çalıştırılır.

A

19

Arayüzler	Soyut Sınıflar
Bir sınıf birden fazla arayüzden türetilebilir.	Bir sınıf sadece tek bir soyut sınıftan türetilebilir.
Sadece boş (gövdesi olmayan) metotlar tanımlanabilir.	Hem normal metot hem de boş metotlar tanımlanabilir.
Çoklu kalıtım özelliği sağlar.	Çoklu kalıtım özelliği sağlamaz.
Tüm ögeler public olarak kabul edilir.	Ögeler public olmak zorunda değildir.
Yapıcı metot içeremez.	Yapıcı metot içerebilir.
Statik ögeler barındıramaz.	Statik ögeler barındırabilir.

B

20

```
interface IGuc
{
void GucAc();
void GucKapat();
}
class Televizyon : IGuc
{
// ...
public void GucAc()
{
}
public void GucKapat()
{
}
}
class Bilgisayar : IGuc
{
// ...
public void GucAc()
{
}
public void GucKapat()
{
}
}
```

4. ÖĞRENME BİRİMİ

A

1	2	3
int[], numaralar	string[],sehirler, string[81]	1

B

4	**5**	**6**	**7**	**8**
B	**A**	**D**	**C**	**E**
9	**10**	**11**	**12**	**13**
B	**B**	**A**	**E**	**D**
14	**15**	**16**	**17**	**18**
B	**A**	**B**	**E**	**D**
19				
C				

5. ÖĞRENME BİRİMİ

A	**1** Arabirim	**2** Application.Run	**3** System.Windows.Forms
	4		
	CenterToScreen		
	FormClosed		
	ControlBox		
	Load		
	AcceptButton		
	Show		
	5 E	**6** A	**7** C
	8 B	**9** D	**10** E

6. ÖĞRENME BİRİMİ

A	**1** D	**2** D	**3** B	**4** E	**5** A
	6 E	**7** E	**8** E	**9** B	**10** D
	11 C	**12** A	**13** B		